Издательство «Знание»

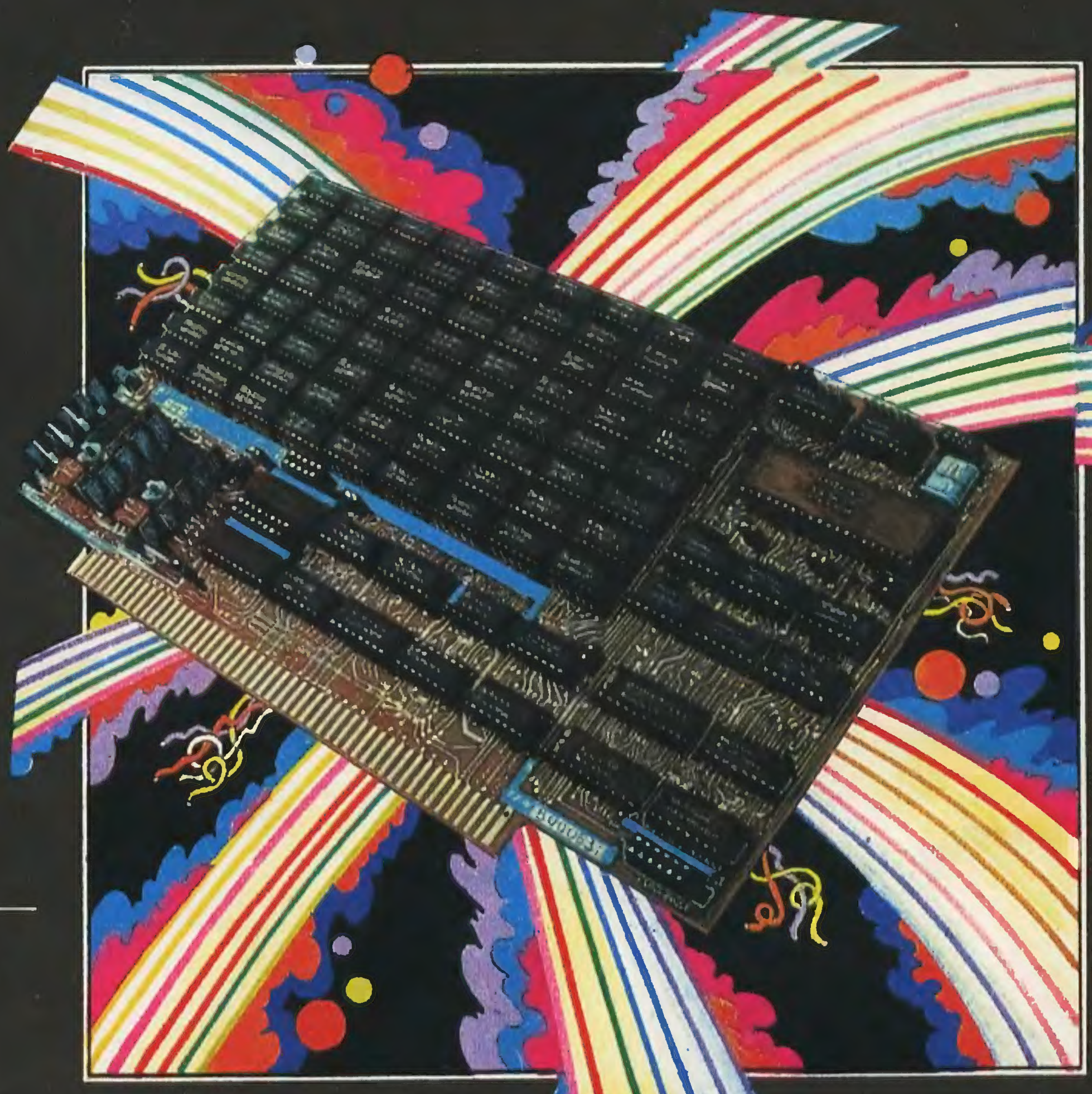
ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА И ЕЁ ПРИМЕНЕНИЕ

Новое в жизни, науке, технике

Подписная научно-популярная серия

Издается ежемесячно с 1988 г.

Микро-процессоры



1989

1

Новое
в жизни,
науке,
технике

ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА И ЕЁ ПРИМЕНЕНИЕ

Подписная
научно-
популярная
серия

1/1989

Издается
ежемесячно
с 1988 г.

МИКРОПРОЦЕССОРЫ

В НОМЕРЕ

3 МИКРОПРОЦЕССОРЫ (история, развитие, технология)

17 Жданов В. С., Горелов В. Ю., Окунев С. Л., Саксонов Е. А. ПРОГРАММИРОВАНИЕ НА ЯЗЫКЕ АССЕМБЛЕРа

30 Горбунов В. Л. ОДНОКРИСТАЛЬНЫЕ МИКРО-ЭВМ

Издательство
«Знание»
Москва
1989

ББК 32.973.2
М59

Авторы выпуска

КУРЕНКОВ Валентин Георгиевич — инженер, специалист по электронике и связи, имеет около 20 публикаций по этой тематике.

ГОРБУНОВ Владимир Леонидович — доцент Московского института электронной техники, лауреат премии им. Л. Живковой, имеет более 30 опубликованных работ, среди них 2 учебных пособия, 20 авторских свидетельств. Награжден двумя серебряными медалями ВДНХ.

ЖДАНОВ Владимир Сергеевич — доктор технических наук, профессор, САКСОНОВ Евгений Александрович — доктор технических наук, ОКУНЕВ Сергей Леонидович — кандидат технических наук, ГОРЕЛОВ Вадим Юрьевич — инженер, работают в Московском институте электронного машиностроения. Специалисты в области создания и разработки микропроцессорных средств.

Редактор Б. М. ВАСИЛЬЕВ

© Издательство «Знание», 1989 г.

Первые годы развития микропроцессорной техники ознаменовались переходом полупроводниковой технологии на новый качественный уровень, а также гигантским ростом производительности, функциональных возможностей и объемов выпуска микропроцессоров (МП). К настоящему времени выпущено огромное количество совершенных микропроцессоров, пригодных для решения практически любой прикладной задачи.

МИКРОПРОЦЕССОРЫ (история, развитие, технология)

Термин «микропроцессор» был впервые употреблен в 1972 г., хотя годом рождения этого прибора следует считать 1971 г., когда фирма Intel выпустила МП серии 4004 — «интегральное микропрограммируемое вычислительное устройство», представляющее собой однокристальный центральный процессор, имеющий в своем составе 4-разрядный параллельный сумматор, 16 4-разрядных регистров, накапливающий сумматор и стек. МП 4004 был реализован на 2300 транзисторах и мог выполнять 45 различных команд. Последующие поколения МП, представляющие собой 8-, 16- и 32-разрядные приборы, появились соответственно в 1972, 1974 и 1981 гг. За прошедшие годы количество транзисторов на кристалле возросло в 200 раз, тактовая частота — в 50 раз, а эквивалентная производительность МП увеличилась на 2—3 порядка величины. По всем показателям темпы микропроцессорной революции не имеют себе равных — они даже превосходят интенсивность развития обычных средств вычислительной техники. Время, прошедшее между изобретением МП и внедрением 32-разрядной архитектуры, составляет 10 лет, т. е. почти вдвое меньше, чем для ЭВМ. Всего лишь за одно десятилетие было ликвидировано значительное отставание микропроцессорной техники от обычных ЭВМ. Более того, микропроцессоры начали выходить на передовые позиции. Сказанное справедливо в отношении не только производительности, но и совершенства архитектуры.

При оценке параметров микропроцессора и выборе микропроцессорной серии наибольшую роль играет разрядность прибора, которая задает элементарный объем обрабатываемых данных. Чем больше разрядность, тем выше производительность и шире возможности адресации. В ранних приборах разрядность регистров, шин управления, а также информационных шин почти всегда была одинаковой. Сейчас такая структура встречается редко. Например, современный микропроцессор Motorola 6800 имеет 32-разрядную внутреннюю архитектуру, 16-разрядную шину данных и 24-разрядную адресную шину (адресует до 16 Мбайт оперативной памяти). Для удобства такую архитектуру называют 32/16/24.

В настоящее время стремятся к большей разрядности, например делают полную 32-разрядную архитектуру (32/32/32).

Если считать, что выпуск предыдущих микропроцессоров должен прекращаться при появлении кристаллов с более высокой разрядностью, то 4-разрядные производились бы всего 1 год, 8-разрядные — 5 лет, 16-разрядные — 5 лет.

Разрядность	Модель	Количество транзисторов, шт	Год выпуска	Торговая марка
4	4004	2200	1971	Intel
8	8008	2300	1972	—«—
8	8080	4800	1973	—«—
8	280	8400	1976	Zilog
8	8048	12400	1977	Intel
16	8086	29000	1978	—«—
16	68000	75000	1980	Motorola
16	80286	130000	1982	Intel
32	NR-9000	450000	1982	Hewlet-Packard

Однако следует оговориться: 4-разрядные микропроцессоры производятся и очень широко применяются до настоящего времени.

Неформальная история[1]

В 1959 г. инженеры из фирмы «Техсас инструментс» придумали, как расположить более одного транзистора на той же самой подложке и соединить эти транзисторы без проводов. Таким образом родилась интегральная схема, или сокращенно ИС. В настоящее время такие тоненькие плоские кусочки кремния могут содержать миллионы транзисторов, их называют попросту чипами.

В 1969 г. фирма, просуществовавшая к тому времени всего один год, называемая «Интел», объявила о создании микросхемы, содержащей 1 Кбит памяти типа RAM. Тогда еще не существовало других микрокомпьютерных чипов, к которым можно было бы привязать эту микросхему памяти, но было много других приложений для такой микросхемы, тем более что эта память была самой емкой из всех тех, что производились до тех пор.

Примерно в то же время — летом 1969 г. — к фирме «Интел» обратилась японская компания «Бьюсиком», занимающаяся выпуском калькуляторов, с заказом на производство набора специализированных микросхем. Микросхемы были разработаны инженерами фирмы «Бьюсиком» для нового семейства калькуляторов этой японской фирмы. Калькуляторы должны были содержать несколько микросхем, каждая из которых состояла из 3000—5000 транзисторов.

Группе инженеров из фирмы «Бьюсиком», которая получила представительство в фирме «Интел», был придан от фирмы «Интел» один из ее разработчиков Марсиан (Тед) Хофф. Хофф познакомился с проектом, предложенным фирмой «Бьюсиком», и нашел, что проект слишком сложен для того, чтобы оказаться рентабельным. До этого он работал с вычислительной машиной PD P-7 фирмы «Диджитал эквипмент корпорейшн», которая имеет очень ограниченный набор команд. Хофф рассудил, что сложность калькулятора может быть сильно уменьшена, если использовать в нем небольшой универсальный процессор. Такой подход, использующий программное обеспечение вместо электронной логики для вычислений, должен резко повысить потребность калькулятора в памяти, но в тот момент фирма «Интел» как раз и занималась производством устройств памяти. Хофф также понял, что такой процессор может быть применен и в других приложениях, и он продал свою идею руководству фирмы «Интел».

Разработчики фирмы «Бьюсиком» все еще продолжали работать по своему первоначальному проекту, когда Хофф и его группа начали работать над альтернативным проектом. Хотя инженеры из фирмы «Бьюсиком» до предела упростили свой проект, все равно каждая микросхема должна была содержать более 2000 транзисторов, и для работающего калькулятора требовалось 12 таких микросхем. Группа Хоффа считала, что для их процессора потребуется 1900 транзисторов.

Универсальный процессор Хоффа выиграл соревнование с проектом фирмы «Бьюсиком», и фирма «Интел» получила от «Бьюсиком» контракт на производство микросхемы универсального процессора, который позже получил название 4004.

4-разрядный микропроцессор

Непосредственное производство микросхем оказалось довольно трудным делом, пока в начале 1970 г. с фирмой «Интел» не начал сотрудничать Федерико Фаджин (который позже основал фирму «Зайлог»). Федерико Фаджин довел микросхему от стадии концепции до кремниевого кристалла всего за девять месяцев. Сначала фирма «Интел» продавала процессор 4004 исключительно фир-

---

[1] Так назвал первую главу своей замечательной статьи «Эволюция микропроцессоров» Марк Герец — сотрудник фирмы «Виасин корпорейшн», на основе которой составлен этот обзор (см.: Byte, — 1985. — № 9). Там, где текст этого реферата идет от первого лица, имеется в виду М. Герец.

ме «Бьюсиком», но летом 1971 г. получила право продавать набор микросхем другим производителям.

В ноябре 1971 г. фирма «Интел» рекламировала свой 4004 как четырехразрядный процессор, выполняющий 60 000 операций в секунду. К февралю 1972 г. фирма «Интел» продала этих процессорных наборов на сумму 85 000 долларов.

Первоначально микропроцессоры выполняли только основные функции, а генераторы синхроимпульсов, запоминающие устройства программ и данных, блоки сопряжения строились на отдельных интегральных схемах. Первые микропроцессоры имели ограниченный набор команд и программировались непосредственно в машинных кодах. Со временем благодаря совершенствованию архитектуры вычислительных систем, появлению языков высокого уровня, росту производительности и гибкости микропроцессорной техники область применения микропроцессоров неизмеримо расширилась. Их используют в промышленности (испытания, управление, контрольно-измерительная аппаратура), в торговле и бытовых приборах. Микропроцессоры ввели в нашу жизнь машинные игры и персональные ЭВМ.

Благодаря включению в 4-разрядные микропроцессоры дополнительных средств интерес к ним не пропадает. Уже реализуются такие их дополнительные средства:

встроенные ПЗУ и ЗУПВ большой емкости;

расширенные наборы команд;

гибкие протоколы ввода/вывода;

встроенные преобразователи;

вспомогательные процессоры;

драйверы дисплеев.

В начале развития микропроцессоры изготовлялись по p-МОП технологии, затем специалисты стали отдавать предпочтение комплементарной МОП-технологии (К/МОП). Теперь применяется множество разнообразных видов технологии и технологических приемов при изготовлении микропроцессоров: n-МОП технология с обогащением и с обеднением, биполярная технология, технология $И^2Л$ и др. Например, за 20 лет в США зарегистрированы 237 технологических нововведений, из них 67 революционных.

Основной материал — кремний, и он «...останется стандартом еще, по крайней мере в ближайшие 10 лет», — считает Джефф Натт, менеджер по вопросам маркетинга процессоров фирмы Motorola. Однако ведущие производители вкладывают большие средства в разработку CaAs-процессоров. Ожидается, что этот материал сыграет ведущую роль при создании новых средств. Если удастся преодолеть технические трудности, то CaAs-процессоры смогут работать в 3 раза быстрее кремниевых, кроме этого они наиболее надежны в условиях повышенной радиации. Исключительная дешевизна и популярность 4-разрядных микропроцессоров обусловлена: во-первых, отработанной для них p-МОП технологией; во-вторых, достигнутой высокой экономической эффективностью их производства из-за стабильного высокого спроса на эти приборы. Другие виды технологий обеспечивают выпуск микропроцессоров по более высокой цене.

Рождение 8-разрядного микропроцессора

В то самое время, когда разрабатывался процессор 4004, фирма CTC («Компьютер текнолоджи корпорейшн», ныне «Дейтаройнт») предложила двум фирмам — «Интел» и «Тексас инструментс» — разработать БИСы для ее нового интеллектуального терминала. Обе фирмы предложили 8-разрядный процессор общего назначения. Можно проследить стереотип развития: 4-разрядный процессор для калькуляторов, поскольку калькуляторы работают в двоично—десятичном коде, и 8-разрядный для терминалов, поскольку терминалы имеют дело с символами.

Фирма CTC не выбрала ни один из этих вариантов; она построила свой терминал на стандартных логи-

ческих микросхемах. Но тем не менее и «Интел» и TI («Тексас инструментс») продолжили работу над своими проектами. «Тексас инструментс» в конце концов получила патент на свою микросхему, а фирма «Интел» дала рождение процессору 8008.

Процессор 8008, представленный в апреле 1972 г., был первым появившимся на рынке 8-разрядным микропроцессором. Он требовал по меньшей мере 20 микросхем поддержки, зато мог выполнять 45 команд со скоростью 300 000 команд в секунду и мог адресовать чудовищное количество памяти в 16 Кбайт. В то время это был огромный объем памяти, и процессор 8008 был серьезным шагом вперед по сравнению с процессором 4004.

Документация по микропроцессорам 4004 и 8008 была сплошной загадкой (по крайней мере так казалось мне; в то время я не имел никакой подготовки по вычислительной технике). Документация подразумевала, что вы знаете все, что было в этой области до того, как вы начали читать эту документацию. Таким недостатком обладает большая часть технической литературы и по сей день.

Первоначальная цель, которую преследовала фирма «Интел», разрабатывая микропроцессоры 4004 и 8008, была замена «произвольной логики» — по-другому говоря, «множеством МИС и СИС, соединенных друг с другом проводами». Очень немногие думали тогда, что эти микросхемы могут пригодиться для универсальных вычислительных машин. Но некоторые фантазеры были заинтригованы самой возможностью иметь вычислительную машину, которая способна делать хоть что-нибудь. Правда, и раньше существовали наборы, из которых можно было собрать вычислительную машину, но они более подходили для демонстрации принципов вычислительной техники, чем для выполнения вычислительных задач. Доступность процессора 8008 радикально изменила ситуацию.

В 1973 г. фирма «Селби компьютер консалтинг инж.» объявила о создании первой универсальной микро-ЭВМ на основе микропроцессора 8008. За этим последовала машина RGS-008 фирмы «RGS электроникс». Затем в июле 1974 г. журнал «Радиоэлектроника» познакомил читателей с микро-ЭВМ Mark-8, разработанной Джонатаном Титусом.

До этого момента все статьи и рекламные объявления о микро-ЭВМ ограничивались публикациями в радиолюбительских изданиях. Mark-8 был первым микрокомпьютером, о котором появилось сообщение в журнале по общим проблемам электроники. Эти первые микрокомпьютеры были все еще более демонстративными, чем полезными, однако микрокомпьютерная революция началась.

Всесильный 8080

В апреле 1974 г. фирма «Интел» заставила нас навсегда изменить сложившееся представление о вычислительных машинах. Фирма объявила о создании микропроцессора 8080 — значительно более совершенного, чем процессор 8008. Микропроцессор 8080 требовал всего шесть микросхем поддержки, выполнял 75 команд, обладал вдесятеро большей пропускной способностью по сравнению с 8008 и адресовал 64 Кбайта памяти. (Ни одна программа, думало большинство в то время, не может быть такой большой!)

Проект микропроцессора 8080 был предложен Фаджином, но группу разработки возглавлял Масатоси Сима, молодой инженер, которого фирма «Интел» переманила из фирмы «Бьюсиком». Учтя неудобства и ограничения процессоров 4004 и 8008, разработчики нового микропроцессора постарались настолько усовершенствовать свой проект, что новая микросхема стала действительно полезным вычислительным элементом. «Интел 8080» — первый микропроцессор, целью создания которого не была простая замена логики. Этот микропроцессор больше походил на вычислительную машину, чем все, что были до него, и с точки зрения аппаратуры был гораздо проще в применении.

В январском 1975 г. номере журнала «Популярная электроника» на первых страницах была помещена первая из серии статей, посвященных построению вычислительной машины «Альтаир 8800», так называемой мини-машины, построенной на основе микропроцессора 8080. Вычислительная машина «Альтаир» была разработана фирмой MIT («Микро инструментейшен энд телеметри системз»), которая была основана Эдом Робертсом для поддержки и обеспечения его экспериментов в области электроники. Весь набор для постройки машины «Альтаир», включая микропроцессор 8080, материнскую плату, блок питания, лицевую панель с множеством лампочек и 256 байтами (не 256 Кбайтами) памяти продавался за 395 долларов.

Сначала все подумали, что это опечатка. Микросхема, содержащая процессор 8080, представленная всего за девять месяцев до того, одна продавалась за 360 долларов. Однако цена была настоящая — фирма MIT вступила с фирмой «Интел» в специальное соглашение. Фирма MIT продала в первый день больше компьютеров, чем надеялась продать за всю жизнь этой разработки.

Вычислительная машина «Альтаир» сыграла значительную роль в том успехе, который сопутствовал микропроцессору 8080, во многом благодаря тому, что теперь программисты имели основания и хорошую отговорку, чтобы писать программное обеспечение для микропроцессорного чипа. Кроме того, разработанная для машины «Альтаир» архитектура открытой шины (улучшенная версия которой стала шиной S-100/ИИЭР 696) позволила всем начать разрабатывать периферийные устройства для этого компьютера.

Одним из таких периферийных устройств был контроллер диска, разработанный фирмой «Диджитал микросистемз», который означал начало использования операционной системы для микропроцессора 8080 и названной CP/M (Управляющая программа для микро-ЭВМ). Операционная система CP/M, детище инструктора Высшей школы военно-морских сил Гарри Килдалла, продавалась за 70 долларов и сыграла основную роль в успехе микропроцессора 8080 и его архитектуры. В результате значительная часть используемого на сегодняшний день программного обеспечения для микро-ЭВМ либо работает на основе набора команд процессора 8080, либо является прямым развитием какой-нибудь системы, которая использовала этот набор команд.

В микропроцессоре 8080 впервые применена n-МОП технология с обогащением, реализована десятичная и ДКД-арифметика, использовалась 16-разрядная адресная шина.

Семейство «Моторола»

В ответ на успех микропроцессора 8080 фирма «Моторола» начала работать над процессором 6800, проект которого был разработан Чаком Педдом. «Моторола» была первой фирмой, которая представила серию микросхем для переферийных устройств, спроектированных специально для работы с данным микропроцессором. Периферийные микросхемы включали устройства для параллельного (микросхема 6820) и последовательного (микросхема 6850) ввода-вывода и обеспечивали включение функций ввода-вывода в систему чрезвычайно простым для разработчиков систем образом.

Разработчики были ориентированы на замену логики микропроцессорами, и требовались миникомпьютеры и кросс-ассемблеры для создания программного обеспечения, а это стоило недешево. К тому времени еще никто не написал ничего, что бы разъяснило, как применять новые микропроцессоры, всем тем, кто ничего не смыслил в их внутренностях или не имел опыта работы с вычислительными машинами.

Фирма «MOS Текнолоджи» опубликовала рекламное объявление, согласно которому фирма собиралась представить на выставке WESCON в сентябре того же года и там же начать продажу 20-долларового микропроцессора. Микропроцессор фирмы «Текнолоджи», получивший наименование 6501,

был полностью совместим по выводам с микросхемой процессора 6800 фирмы «Моторола» — вы могли выпаять из платы микропроцессор 6800, вставить на его место 6501, и он начнет работать. Правда, программное обеспечение все равно пришлось бы поменять, поскольку существовали определенные различия в архитектуре микропроцессоров и в их наборах команд. Фирма «MOS Текнолоджи» также планировала выпуск версии микропроцессора со встроенной схемой тактирования, наличие которой на отдельной микросхеме требовали все ранее выпускавшиеся микропроцессоры. Эта версия микропроцессора должна была получить наименование 6502 и стоить 25 долларов.

Все производители микропроцессоров пришли в волнение. В то время процессоры 8080 фирмы «Интел» и 6800 фирмы «Моторола» продавались по розничной цене 179 долларов за штуку. М. Герец вспоминает: «Прекрасно помню, как я стоял в вестибюле (т. е., собственно, в гостиной) фирмы «Е-Ми системз» со Скоттом Уэджем и Дейвом Россумом, которые как раз незадолго до этого начали встраивать микропроцессор 8080 в свои синтезаторы. Мы разговаривали с коммерческим представителем фирмы «Интел», который считал рекламное объявление просто мистификацией или трюком. Он сказал, что руководство фирмы «Интел» просто не может производить микропроцессоры с такой ценой, и либо рекламное объявление было эффектным трюком, либо речь шла об оптовой цене — за миллионы штук. Я сказал, что не может быть, чтобы микропроцессорные чипы шли по тому же пути, что и микросхемы для научных калькуляторов — сначала сотни долларов, а теперь всего несколько долларов. Коммерческий представитель фирмы «Интел» сказал, что микропроцессоры никогда не будут дешевле 100 долларов.

Точка зрения коммерческого представителя фирмы «Интел» была почти всеобщей в Кремниевой Долине, но «MOS Текнолоджи» была расположена на Восточном побережье. Я позвонил туда, и они подтвердили, что это действительная цена и это цена за одну-единственную штуку, и что микросхемы можно купить в их апартаментах в соседнем отеле.

Я пошел туда в тот же вечер и увидел, что апартаменты фирмы заполнены народом. Микросхемы были насыпаны в два огромных круглых аквариума.

Фирма «MOS Текнолоджи» тогда же в отеле вечером продемонстрировала еще одну новинку — первые многофункциональные периферийные микросхемы. Фирма поместила память типа RAM, память типа ROM, таймер и ввод-вывод в одну микросхему. Одна версия называлась TIM (монитор терминального интерфейса) и содержала в одной микросхеме полный монитор для общения с серийным терминалом. Вторая схема называлась KTM (монитор клавишного ввода) и была встроена в микрокомпьтерную плату, которая имела клавиатуру, центральный процессор, дисплей, память типа ROM, память типа RAM, параллельный ввод-вывод и стоила 245 долларов. Это уже была полная система. Ни один поставщик микропроцессоров не предлагал ничего подобного раньше. Это было впечатляюще!

В тот вечер я возвращался домой, унося в качестве добычи мой 25-долларовый микропроцессор и два руководства. В тот момент я не отдавал себе отчета в том, что мне придется потратить еще 300 долларов, чтобы создать работоспособную систему, не считая используемого в качестве терминала устройства «ASR-33 Телетайп».

В тот первый день ярмарки WESCON фирмы «Интел» и «Моторола» снизили цену на свои микросхемы до 69 долларов 95 центов. Революция была на полном ходу, электронная промышленность уже не могла повернуть назад».

Микропроцессору 6501 была суждена короткая жизнь. Фирма «Моторола» возбудила против фирмы «MOS Текнолоджи» дело, обвинив последнюю в том, что Чак Педдл украл технологию у фирмы «Моторола» и что микропроцессор 6501 создан с нару-

шением авторских прав, поскольку был взаимозаменяемым со своим аналогом «Моторола 6800». Фирма «MOS Текнолоджи» согласилась прекратить выпуск процессора 6501.

Однако многие вычислительные машины, построенные на основе микропроцессора 6502, стали теперь легендой: собственная разработка фирмы «MOS Текнолоджи» KIM-1, машины «Apple I» и II, различные модели «Атари», а также среди прочих машины «Коммодор PET» и VIC-20.

Появление Z80

Где-то в конце 1975 или в начале 1976 г. Федерико Фаджин покинул фирму «Интел» и основал новую фирму под названием «Зайлог». В 1976 г. фирма «Зайлог» объявила о создании микропроцессора Z80, значительно улучшенного варианта микропроцессора 8080, который включал весь набор команд последнего и соответственно позволял работать матобеспечению, написанному для микропроцессора 8080. Фирма также объявила, что имеются микропроцессоры, которые работают на тактовой частоте 4 МГц, т. е. вдвое большей, чем процессор 8080. Кроме того, микропроцессор Z80 выполнял множество дополнительных команд — общее количество команд равнялось 176. Это было слишком хорошо, чтобы казаться правдой, и микропроцессор Z80 был принят с тем же скептицизмом, что и 6502 незадолго до этого. Однако микропроцессор Z80 был реальностью, и действительно имелись партии процессоров, которые работали на частоте 4 МГц.

Несколько плат для набора Z 80 с шиной S-100 появились на рынке тут же после разработки микропроцессора, и всем сразу понадобились эти платы. Тем не менее, несмотря на то что микропроцессор Z 80 был значительно мощнее процессора 8080 в смысле набора команд, очень немногие отважились писать программы с учетом преимуществ дополнительных команд Z 80. Причина здесь простая: большинство машин, работавших в то время, были созданы на основе микропроцессора 8080, и если вы стали бы писать программы, которые могут работать только на микропроцессоре Z 80, то ваш рынок оказался бы слишком маленьким. Это обстоятельство преследует микропроцессор Z 80 и по сей день.

Тем не менее разработчики не стали применять микропроцессор 8080 в новых системах. Микросхема, содержащая процессор Z 80, была несравнимо более легкой в использовании, для нее требовалось только одно напряжение питания, и при этом не нужны микросхемы поддержки. И этот микропроцессор работал гораздо быстрее, даже без использования дополнительных команд.

Процессор Z 80 быстро приобрел завидную популярность. В настоящее время он продается в розничной торговле по цене менее 1 долл. за штуку (55—65 коп.). Уместно напомнить, что в 1975 г. представитель фирмы Intel в ответ на слухи о появлении процессора 6502 стоимостью в 25 долл. утверждал, что никогда микропроцессоры не будут стоить меньше 100 долл. за штуку, т. е. скепсис в отношении цен на микропроцессоры носил в то время массовый характер.

Вместе с микропроцессором Z 80 пришел еще один новый принцип микрокомпьютерных схем: встроенная схема регенерации динамической памяти RAM. Динамические устройства памяти типа RAM всегда стоили примерно вчетверо дешевле своих статистических конкурентов в расчете на один бит, и это делало их очень привлекательными для использования. Однако из-за того что приходится постоянно регенерировать такую память (иначе она забывает свои данные), ее использование было чрезвычайно сложным делом. Наличие в микропроцессоре Z 80 схемы регенерации динамической памяти было очень большим шагом вперед и позволяло строить системы, которые были гораздо дешевле предыдущих. Такие микро-ЭВМ, как TRS-80 фирмы «Радио шак», разработанная Стивом Ленингером, использовали именно это преимущество микропроцессора.

Одновременно оптимизировать все характеристики микропроцессора нельзя, и стали изготовлять целые микропроцессорные семейства с еди-

ной архитектурой. Каждое семейство имеет свою собственную комбинацию ПЗУ, ЗУПВ и дополнительные функциональные возможности.

Новейшие 8-разрядные процессоры представляют собой упрощенные варианты широко распространенных 16- и 32-разрядных приборов.

16-разрядные микропроцессоры

Тем временем фирма «Интел» признала, что ее микропроцессор 8080 нуждался в модернизации. В 1976 г. фирма объявила о создании процессора 8085. Этот микропроцессор выполнял все команды 8080-го и еще несколько дополнительных команд. Однако «Зайлог» застал «Интел» врасплох. Те преимущества, которые имел процессор 8085 перед 8080, не шли ни в какое сравнение с мощью микропроцессора Z 80. С чисто аппаратной точки зрения процессор 8080 был намного приятнее, чем Z 80, но последний имел неоспоримое преимущество — он был быстрее. Микропроцессор 8085 работал на частоте 3 МГц, в то время как фирма «Зайлог» уже продавала свои микропроцессоры, которые работали на тактовой частоте 4 МГц. Фирме «Интел» нужно было предложить в ответ что-то совсем новое — таким ответом оказался 16-разрядный микропроцессор.

Попытки создания 16-разрадных микропроцессоров делались и раньше. Фирма «Нейшнл семикондактор» начала работу по созданию микропроцессорного набора IMP-16 еще в 1972 г. Соответствующий компьютер был действительно построен, но так и не появился на рынке. Билл Гадбаут сказал, что рынок не готов к появлению 16-разрядной микро-ЭВМ и, по всей видимости, он был прав.

В литературе некоторые авторы упоминают о создании в 1974 г. фирмой National Semiconductor первого 16-разрядного микропроцессора PACE. А в 1976 г. разработан еще один представитель этого поколения — TMS 9900 фирмы Texas Instruments. Сейчас имеется несколько версий исходных микропроцессоров (к ним относятся также МП 8086, 80000 и МС 68000), обладающих повышенной производительностью. Трудно дать подробный анализ всех известных сегодня 16-разрядных микропроцессоров, и поэтому далее будет рассказано только о наиболее популярных.

Слухи о существовании партии 16-разрядных микропроцессоров ходили годами, но сами микропроцессоры не появлялись до декабря 1983 г., когда фирма «Вестерн дизайн сентер» (не путать с «Вестерн диджитал») объявила о создании 16-разрядной версии микропроцессора 6502.

Итак, возвращаясь к прерванному повествованию, отметим, что фирма «Интел» нуждалась в чем-то, что могло бы конкурировать с микропроцессором Z 80. Фирма пришла к выводу, что большинство предыдущих 16-разрядных разработок окончилось неудачей из-за того, что не существовало программного обеспечения, которое могло бы быть легко перенесено на эти микропроцессоры — все это были первые микропроцессоры в жизни выпускавших их фирм.

«Интел», однако, имел серьезное преимущество: растущую базу программного обеспечения для микропроцессора 8080. Фирма решила, что ее 16-разрядный микропроцессор будет прямым развитием процессора 8080. К сожалению, разработчики не сохранили полную совместимость с программами для 8080, но по крайней мере каждый регистр микропроцессора 8080 имел соответствующий регистр в процессоре 8086, что делало возможным создание трансляторов программ 8080 — 8086 и соответственно давало программистам привычную для них отправную точку. Это сыграло большую роль в успехе микропроцессора 8086, который был представлен миру в 1978 г.

А затем кого-то в фирме «Интел» осенила догадка. Почему бы не сделать аппаратную часть также легко переносимой, как и программное обеспечение? Так родился микропроцессор 8088. Микропроцессор 8088 — это 8086 внутри, но снаружи 8-разрядная шина данных. Когда процессору

требуется выбрать 16-разрядное данное, он сначала получает один байт, а затем второй. Программист не должен заботиться об этом, все происходит автоматически на аппаратном уровне. Выходные сигналы микропроцессора 8088 выглядят как сигналы 8085-го процессора, к которому разработчики микропроцессорной техники уже привыкли. Таким образом, было очень легко расширить имеющиеся 8-разрядные проекты до 16-разрядных.

Вы можете подумать, что из-за такой процедуры, обеспечивающей совместимость, процессор будет работать вдвое менее эффективно, чем он может. Но разработчики из фирмы «Интел» учли это при проектировании 8086. Этот микропроцессор внутри состоит из двух различных, но связанных процессоров. Один из этих процессоров имеет исполнительное устройство, которое, собственно, и выполняет обработку данных или выполняет команды. Другой процессор получил название «устройство шинного интерфейса» (BIU). Устройство шинного интерфейса выполняет все действия по связям с внешним миром, а также на нем лежит ответственность за генерирование адресов, выдачу данных в систему и получение данных из системы. Внутри BIU имеется очередь. В то время когда исполнительное устройство занято перемалыванием данных, устройство шинного интерфейса получает с шины следующую команду и ставит ее в очередь. Устройство шинного интерфейса микропроцессора 8086 способно идти впереди исполнительного устройства на 6 байт, храня эти байты в своей очереди. Благодаря наличию очереди быстродействие процессора 8088 лишь на 20% (в среднем) ниже, чем у процессора 8086. (См. статью «Основные сведения о микропроцессорах 8086 и 8088 фирмы «Интел», журнал «Байт», июль 1983, стр. 14.) Микропроцессоры 8086/8088 были первыми, использовавшими механизм очередей.

Фирма «Интел» ввела еще одну новинку в свое семейство микропроцессоров 8086 — наличие сопроцессоров. Суть этого принципа состоит в подвешивании непосредственно на шину главного процессора другого процессора, расширяющего функции главного. Наиболее примечательным из таких сопроцессоров был процессор 8087, математический сопроцессор, который добавлял в набор команд процессоров 8086/8088 был предназначен исключительно для выполнения арифметических операций, он выполнял их чрезвычайно быстро.

Улучшенный вариант МП 8086 — это МП 80286, выпущенный в 1982 г. Это полностью совместимые приборы, как говорят, совместимы «снизу вверх». Такая совместимость препятствует внесению крупных изменений в архитектуру нового прибора и, соответственно, ограничивает реализацию широкого набора новых функций и типов данных. В МП 80286 при 5-кратном увеличении числа транзисторов был достигнут 6-кратный рост производительности. На одном кристалле площадью 43,9 мм^2 размещены 130 тыс. транзисторов, что позволило добавить еще схемы управления памятью и 4-уровневую систему ее защиты. МП 80286 обладал виртуальной памятью, значительно расширенной системой команд.

Микропроцессор MC68000

В 1977 г. разработчики из фирмы «Моторола» занимались созданием нового процессора для рынка 16-разрядных микропроцессоров, обещая при этом сделать его внутри 32-разрядным. Они также поставили своей целью исключить все специализированные команды и позволить процессору выполнять все операции, на всех регистрах, над всеми типами данных и со всеми способами адресации. Такое свойство называется ортогональностью. Программистам нравится ортогональность, поскольку она означает, что им не нужно запоминать всякие исключения в наборе команд. Результатом усилий разработчиков фирмы «Моторола» было появление в 1979 г. микропроцессора MC 68000.

Микропроцессор MC 68000 — это как раз один из таких процессоров,

которые одни любят, а другие терпеть не могут. Равнодушных здесь очень мало. По сравнению с семейством 8086/8088 этот микропроцессор требовал огромных программистских усилий, прежде чем он смог бы делать хоть что-то. Однако становилась популярной операционная система UNIX, а процессор 68000 выглядел как неплохо приспособленная для UNIX машина. Было представлено несколько машин на базе микропроцессора 68000 с операционной системой UNIX, но пожалуй, ни одна из них не имела успеха.

Фирма «Моторола» также объявила о своем микропроцессоре 68008, 8-разрядной версии процессора 68000, сходной по концепции с процессором 8088. Однако процессор 68000 не имел настоящего механизма очереди, а это означало, что процессор 68008 работал вдвое медленнее, чем 68000. Конечно, процессор 8088 выглядел на этом фоне предпочтительнее.

По своей внутренней архитектуре процессор 68000 был схемой с микропрограммным управлением, т. е. его внутренние функциональные элементы были универсальными. В такой архитектуре память типа ROM (которая хранит микропрограмму) управляет работой микросхемы. Реакция процессора на каждую команду управляется микропрограммной памятью. Если команда выполняется неправильно, вы можете исправить ее прямо в памяти микропрограммы, и до известных пределов вы можете даже модифицировать набор команд, если хотите. (Дополнительный процессор фирмы IBM для ее системы XT/370 есть на самом деле микропроцессор 68000 со специализированной микропрограммой.)

Вплоть до описываемого времени все процессоры были обычными устройствами с произвольной логикой, которые имели внутри центрального процессора маленькие схемки для выполнения каждой отдельной операции. Оба подхода обладают как преимуществами, так и недостатками — микропрограммное управление дает процессору определенную гибкость за счет потери в быстродействии, в то время как произвольная логика дает быстродействие за счет потери гибкости. Кроме того, исправлять ошибки в системах с произвольной логикой чрезвычайно тяжело — особенно когда разработчик покидает фирму.

В фирме «Зайлог», например, Масатоси Сима начал работу над 16-разрядным процессором, использующим произвольную логику. Принцип произвольной логики отлично работал в микропроцессорах 8080 и Z80; однако то, что при проектировании 16-разрядного микропроцессора Z8000 Сима пошел по тому же пути, сыграло с фирмой «Зайлог» злую шутку. После того как первый кристалл с микропроцессором Z8000 был сделан, но до того, как его группа закончила вылавливание всех ошибок в схеме, Сима покинул «Зайлог», чтобы вернуться в «Интел». Фирма «Зайлог» так и не смогла выудить все ошибки из схемы процессора Z8000. Кроме того, фирма «Зайлог» подала в свое время хороший пример, обеспечив для микропроцессора Z80 полную совместимость со всем программным обеспечением, разработанным для предыдущего поколения 8080. К сожалению, на этот раз фирма не последовала своему же собственному примеру и создала процессор Z8000 совершенно непохожим на процессор Z80. Фирма «Зайлог» сделала попытку вместить целый мини-компьютер на один чип, но к сожалению, работу эту сделала не очень хорошо. Различие в системах команд с микропроцессором Z80, невыловленные до конца ошибки в схеме и недостаточно мощные команды из-за принципа произвольной логики, положенного в основу разработки, — все это сыграло свою роль в том, что микропроцессор Z8000 так и не имел успеха.

В 1981 г. фирма «Нейшнл семикондактор» сделала вторую попытку выпустить на рынок 16-разрядных микропроцессоров свой процессор, на этот раз 16032. Микропроцессор 16032 должен был стать 32-разрядным микропроцессором (по внутренней шине данных) с 16-разрядной

внешней шиной. Так как фирма «Моторола» так и не сделала своего обещанного арифметического сопроцессора, а фирма «Интел» так и не смогла преодолеть 5-мегагерцевый барьер своего арифметического сопроцессора 8087, все были поражены, когда фирма «Нейшнл семикондактор» объявила о создании своего арифметического сопроцессора, который работал на частоте 10 МГц, обеспечивая вдвое большую производительность, чем процессор 8087 фирмы «Интел».

К сожалению, фирма «Нейшнл» стала первой компанией из производителей микропроцессоров, которая начала продавать абсолютно готовые периферийные микросхемы до того, как стал доступным сам микропроцессор. До сегодняшнего дня процессор 16032 еще имеет ошибки в схеме. Тем не менее программистам нравится система команд этого микропроцессора, потому что она напоминает им систему команд машин VAX (семейство высокопроизводительных супер-мини-компьютеров фирмы «Диджатал эквипмент»). Поскольку машины семейства VAX оказались столь удачными в качестве носителей операционной системы UNIX, микропроцессор 16032 тоже должен был бы иметь успех, как основа для компьютера с операционной системой UNIX. И сейчас это обстоятельство придает микропроцессору 16032 жизненную силу.

В 1982 г. фирма «Интел» опять подтолкнула вперед полупроводниковую промышленность, объявив о создании процессора iAPX 286. Архитектура этого микропроцессора была значительной модернизацией процессора 8086, включавшей встроенное управление виртуальной памятью и многие другие особенности, разработанные специально для использования в многозадачных, многопользовательских приложениях. Новый микропроцессор имел режим, в котором он работал прямо как процессор 8086, и фирме удалось добиться существенного увеличения быстродействия в этом режиме. Встроенное управление памятью — это было то, чего ни один микропроцессор не имел в то время. Главное преимущество встроенного контроля памяти состоит в том, что такое управление работает намного быстрее, чем в процессорах, требующих внешнего управления памятью. Использование микропроцессора 80286 фирмой IBM в ее персональной ЭВМ IBM PC AT обеспечило этому микропроцессору успех по крайней мере на следующие несколько лет.

32-разрядные микропроцессоры

Уже в 1980 г. начали выпускать микропроцессоры, у которых внутренние шины были 32-разрядными, а внешние 16-разрядными. Однако эра полных 32-разрядных приборов началась только в 1981 г., когда на рынке появился МП iARX 432 фирмы Intel.

Несмотря на проблемы, которыми сопровождался выпуск процессоров фирмы «Нейшнл семикондактор» была первой фирмой, объявившей о создании и начавшей реально продавать настоящие 32-разрядные микропроцессоры. Ее микропроцессор 32032, как и в свое время 16032, программно совместим с 32016. Еще рано говорить о том, какая жизнь суждена процессору 32032, но популярность операционной системы UNIX, возможно, станет основным фактором успеха этого микропроцессора, если только на самом деле все обещания относительно того, что семейство микропроцессоров 32000 является идеальным для операционной системы UNIX, окажутся правдой.

Фирма «Моторола» в настоящее время выпускает пробные образцы своего нового микропроцессора, являющегося полностью 32-разрядным расширением процессора 68000,— процессора 68020, который выглядит многообещающе. Этот микропроцессор имеет одну особенность, которая, возможно, позволит сразу использовать его в большом количестве разработок. Вспомним, что процессор 8088 оказался столь успешным бла-

годаря наличию 8-разрядной внешней шины и соответственно легкости применения в самой различной аппаратурной архитектуре. Микропроцессор 68020 позволяет нам динамически выбирать размер шины 8, 16 или 32 разряда по желанию. Предположительно этот микропроцессор будет совместим на программном уровне с процессором 68000 и будет быстродействующим. Одним из факторов, обеспечивающих его высокое быстродействие, является наличие кэш-памяти — логическое продолжение принципа очереди, использованного с большим успехом в микропроцессорах 8086/8088/80286.

Кэш-память в процессоре 68020 имеет объем 256 байт и работает по несколько иному принципу, чем очередь. Если выполняется команда перехода на адрес, находящийся в очереди, вся очередь сбрасывается и перезагружается. В противоположность этому кэш работает в точности как память, и поэтому переход на адрес внутри кэша не означает его сброса и перезагрузки. Если цикл оказывается достаточно маленьким, то он может работать непосредственно из кэш-памяти. Преимущество состоит в том, что доступ процессора к кэш-памяти происходит гораздо быстрее, чем к внешней памяти. Таким образом, программы работают быстрее, если они хранятся в кэш-памяти. Как и в случае процессора 32032, рано еще говорить об окончательном успехе процессора 68020.

Современные 32-разрядные микропроцессоры производятся по КМОП и НМОП технологиям, производительность их 1—5 млн. оп/с и готовится производство процессоров с производительностью 15—20 млн. оп/с.

Стандартная фоннеймановская архитектура ЭВМ уступает место так называемой гарвардской архитектуре, отличающейся параллельным выполнением операций и соответственно более высокой производительностью. Разработаны кристаллы с новой так называемой RISC-архитектурой, имеющие производительность 4—10 млн. оп/с. Однако независимо от архитектурных различий 32-разрядные процессоры должны взаимодействовать с 16- и 8-разрядными периферийными устройствами, и для этого они обладают возможностью динамически изменять формат магистралей.

В настоящее время около десяти фирм поставляют традиционные и упрощенные по системе команд (RISC-архитектура) 32-разрядные микропроцессоры, а четыре фирмы предлагают микропрограммируемые 32-разрядные кристаллы.

RISC-процессоры программируются меньшим и упрощенным набором команд. Соответственно они часто обгоняют своих более именитых конкурентов за счет того, что успевают выполнить команду за каждый тактовый цикл процессора. Однако скептики не относят это к достоинству, указывая, что RISC-процессоры переносят бремя обработки информации на программное обеспечение — операционную систему и прикладные программы, но последние значительно отстают пока от аппаратной части.

Производительность компьютеров с архитектурой RISC примерно в 5 раз выше, чем для традиционной архитектуры. Для выполнения каждой команды такому компьютеру требуется ровно один такт. Для сравнения: микропроцессору MC 68000 на это нужно пять тактов.

Некоторые специалисты считают, что, даже если микропроцессоры с архитектурой RISC не будут иметь коммерческого успеха, исследования, связанные с их разработкой, могут дать мощный импульс для дальнейшего развития микропроцессоров с традиционной архитектурой.

Фирма «Зайлог» объявила о выпуске процессора под названием Z800, 16-разрядного расширения процессора Z80, который должен быть полностью совместим с последним на программном уровне. Однако работоспособный кристалл так и не получился. Вместо этого фирма пытается заставить свой микропроцессор Z80 работать на частоте 10 МГц. Фирма также объявила о создании процессора

Z80000, 32-разрядной версии процессора Z80006, который я называю процессором типа кухонной раковины, поскольку этот процессор хочет иметь все, включая программную совместимость с Z80006, что само по себе неплохо, но процессор получился плохой.

RISC-компьютер был успешно реализован в кремнии в Беркли, и что удивительно — с первой попытки. В настоящее время создатели этой микросхемы работают над увеличением ее быстродействия. Этот процессор имеет 32-разрядную архитектуру.

И наконец, следует сказать об уникальном устройстве, которое получило название «Транспьютер». Этот микропроцессор предназначен для параллельной обработки. Многие годы еще займет написание программного обеспечения для этого процессора, пока удастся получить полную отдачу от него, но теоретические возможности транспьютера просто поражают воображение.

По мере перехода к параллельной архитектуре будут разработаны специальные параллельные процессоры и программные языки, которые еще больше увеличат скорость и эффективность такой архитектуры. Об этом свидетельствует транспьютер компании Inmos — микропроцессор, использованный в супер-компьютерах серии «T» фирмы «Floating Point Systems», имеющий потрясающую скорость — 10 MIPS (MIPS — 1 млн. одноадресных операций в секунду).

Такую же скорость без особых затруднений — так заявил менеджер фирмы Intel Сайва Кумара, — получит на своих микропроцессорах фирма Intel в ближайшие несколько лет. К 2000 г. «мозг», управляющий компьютером, будет отдавать команды со скоростью, которая достигнет 30 MIPS — почти половины скорости супер-компьютера Cray-1.

А между тем микропроцессор 4004 до сих пор управляет светофором неподалеку от моей конторы. Похоже, что в будущем он заткнет за пояс любые сверхумные новинки, не правда ли?

Что за 32-разрядными!

По логике развития за 32-разрядными микропроцессорами следует ожидать 64-разрядные. Однако пользователи и разработчики пока еще не давали заявки на какие-либо системы свыше 32-разрядов. Например, большие универсальные ЭВМ уже более 35 лет как остановились в своем развитии на 32 разрядах, и нет никаких видимых признаков продвижения к 64-разрядной архитектуре. Хотя вполне возможны заявки на виртуальное адресное пространство объемом более 2^{32} байт. Фактически такая потребность удовлетворяется и с помощью 32-разрядных процессоров с сегментированной памятью (например, в модели 386 обеспечивается пространство виртуальных адресов размером 2^{46} байт).

Можно ожидать от перспективных процессоров наличия в них 64-разрядных информационных каналов, что позволит увеличить их производительность. Применительно к шинам в ближайшем будущем не ожидается потребности в физических адресах более 32-разрядов, но могут появиться 64-разрядные шины между процессорами и памятью.

Реферат подготовил
В. Г. Куренков

СТАТЬЮ ПИШЕТ КОМПЬЮТЕР[1]

Кого удивишь сегодня в ФРГ компьютером, если они сопровождают тебя здесь, что называется, с пеленок? Их встречаешь в детских садах и школах, в любых, даже самых захудалых магазинах, на скромных заправочных станциях, в редакциях газет и личных бюро журналистов...

И все-таки это сообщение заинтриговало многих, в том чис-

[1] Литературная газета. — 1988. — № 34 (5204) — 24 августа. — С. 9

Продолжение см. на стр. 16

ле и вашего корреспондента. Сенсационная новость поступила из федерального общества по вопросам математики и обработки информационных данных (ОМД) в Санкт-Аугустине под Бонном. Речь идет о совершенно новом виде компьютера, модели XXI века — компьютере-«ассистенте».

— Кофе он, конечно, пока (!) сварить не сможет, но его прообразом является человек, точнее — секретарь-ассистент, — восторженно делился со мной один из знакомых в федеральном министерстве научных исследований.

С этим не согласны, однако, авторы привлекшего всеобщее внимание проекта.

— Сравнение с человеком опасно, — заявил мне один из них, доктор Дикхофф. — Мы не пытаемся вытеснить из бюро человека, а хотим помочь ему избавиться от множества примитивных операций.

Добавляю от себя, что эти разъяснения последовали из Санкт-Аугустина после появле-

ния в печати опасений, что новые компьютеры увеличат и без того громадную армию лишних людей в ФРГ. Видимо, основания для тревог есть, ибо сам доктор Дикхофф признал, что в отдельных случаях фирмачи смогут сократить свои штаты, а те, у кого нет средств на содержание секретаря, обойдутся без него.

Новый компьютер будет во много раз умнее самых современных персональных компьютеров. Он сможет выполнять до тысячи операций. «Ассистент» в состоянии даже корректировать сумбурные указания своего хозяина-человека, если тот чем-то взволнован либо просто ошибся. Как опытная секретарша, он поможет шефу добрым советом, а ученым укажет путь из тупика к верному решению.

— Наш «ассистент» способен даже проявлять инициативу. Для этого он будет обладать поистине энциклопедическими знаниями, чтобы, например, предупредить человека, когда достигнуты границы возможных на сегодня решений, — рассказывали мне в Санкт-Аугустине.

Авторами предусмотрены три типа компьютеров.

«Универсальный ассистент», который поможет не только подобрать материалы для написания статьи или доклада, но и будет их соавтором.

«Ассистент-специалист», предусмотренный для обслуживания конкретных профессий. Это прежде всего компьютер-статистик, который также сможет готовить своему шефу проекты документов.

Наконец, «ассистент-коммуникатор», он ближе всех к нынешней секретарше. В его обязанности будет входить поддержание контактов с клиентами, прием и ответы на телефонные звонки, письма, телексы. Этот помощник сможет отобрать самые важные документы и направить в корзину все лишнее, не представляющее интереса для данной фирмы или человека.

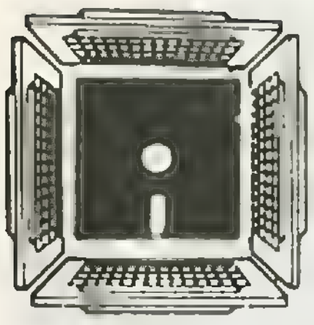
Фантастика? Возможно ли все это? Когда?

— Мы уверены в своем проекте. Надеемся, что первые компьютеры — «ассистенты» появятся в бюро уже в начале 90-х годов, а их массовое исполнение начнется на пороге 2000 года, — считает доктор Дикхофф. — Кстати, стоимость нового компьютера существенно не возрастет.

Нам пишут

Уважаемая редакция! Очень прошу вас подробно рассказать в ваших выпусках о программировании на языке АССЕМБЛЕРа для микрокалькулятора КР580ИК80А.

г. Ленинград, Герасимов П. Р.



Вы хотите поговорить с компьютером! А вы знаете, что для этого надо сделать! А надо то же, что и для разговора с иностранцем: либо выучить его язык, либо обратиться к переводчику. Переводчиком при общении с ЭВМ служит транслятор — специальная программа, позволяющая ваши задачи сделать понятными компьютеру. Однако свое обращение к ЭВМ вам придется излагать не простым языком, а алгоритмическим.

ПРОГРАММИРОВАНИЕ НА ЯЗЫКЕ АССЕМБЛЕРа

ЖДАНОВ В. С.,
ГОРЕЛОВ В. Ю.,
ОКУНЕВ С. Л.,
САКСОНОВ Е. А.

Введение

Существует много различных алгоритмических языков высокого уровня. Вы, наверное, слышали о таких из них, как ФОРТРАН, АЛГОЛ, ПАСКАЛЬ. Они отличаются от нашего разговорного языка, но достаточно удобны и просты, позволяют описать ход решения задачи. Выучив один из таких языков, вы сможете «общаться» с ЭВМ самых разных типов, если для данного компьютера разработан соответствующий транслятор. Транслятор «разъяснит» компьютеру, что вы от него требуете, и компьютер выполнит вашу программу. Программируя на языке высокого уровня, вы далеки от ЭВМ, не чувствуете ее особенностей, не до конца осознаете ее возможности. «Виноват» в этом слишком «угодливый» переводчик, который скрывает от вас все, что происходит в ЭВМ, и готов сделать все, чтобы никаких дополнительных знаний, кроме алгоритмического языка, вам для «разговора» с ЭВМ не понадобилось. Конечно, приятна такая забота, но в этом случае для вас возможности алгоритмического языка и возможности ЭВМ почти одно и то же, а что же происходит внутри ЭВМ, как она «думает», какими словами «мыслит», решая вашу задачу, остается для вас загадкой. А где есть загадка, там всегда есть стремление разгадать ее. Что же поможет вам понять «мысли» ЭВМ? Есть ли такой переводчик, который не только компьютеру расскажет о вашей задаче, но и объяснит вам, как компьютер сможет решить эту задачу? Есть такой переводчик — это АССЕМБЛЕР. Точно определить, что такое АССЕМБЛЕР, достаточно трудно. Ассемблер — это программа, которая позволяет перевести программу, написанную на специальном языке, на язык, понятный ЭВМ. Ассемблер не обещает вам легкой жизни, так как, кроме языка, понятного ассемблеру, вам необходимо знать о том, как представляется информация в ЭВМ, что входит в состав ЭВМ. Зная все это, вы, однако, не сможете легко переходить с одного типа ЭВМ на другой, так как ассемблер и его язык машинозависимы, т. е. ориентированы на ЭВМ определенного типа; для каждого типа ЭВМ имеется свой ассемблер. Язык АССЕМБЛЕРа ЭВМ того или иного типа нередко называют просто АССЕМБЛЕР. Чтобы разделить эти два понятия, в случае, когда речь идет о программе, будем писать ассемблер, а вместо слов «язык ассемблера» — АССЕМБЛЕР. Изучив язык АССЕМБЛЕР хотя бы одного компьютера, вы перестанете с внутренней робостью относиться к компьютеру, его «мысли» станут вам доступны и понятны. Знание АССЕМБЛЕРа позволяет пол-

но и эффективно использовать технические возможности ЭВМ, создавать самые быстродействующие и компактные программы. В ряде случаев, особенно при работе с микро-ЭВМ, АССЕМБЛЕР является одним из важнейших средств программирования, и некоторые задачи наиболее просто программируются именно на этом языке. Знание АССЕМБЛЕРа позволяет не только понять, как «мыслит» компьютер, но и показывает, каким образом один и тот же микропроцессор может управлять и автомобилем, и станком, и стиральной машиной.

Итак, чтобы двигаться дальше, нам нужно разобраться с двумя вопросами: что входит в состав ЭВМ и какой язык будет «родным» для ЭВМ?

Структура ЭВМ

Чтобы понять, как «мыслит» ЭВМ и что такое АССЕМБЛЕР, надо сначала узнать, что же такое ЭВМ, из чего она состоит. ЭВМ различаются по своим функциональным возможностям, внешнему виду, габаритам, однако какими бы ни были отличия, все машины роднит их общая внутренняя структура. Любая современная ЭВМ содержит процессор, память и систему ввода-вывода. Процессор — это основной элемент ЭВМ (центральная ее часть), осуществляющий процесс обработки данных и управляющий этим процессом. Процессор дешифрирует (т. е. распознает) и выполняет команды программы, организуя обращения к оперативной памяти и через систему ввода-вывода к внешним устройствам ЭВМ. Процессор осуществляет управление взаимодействием всех устройств, входящих в состав ЭВМ. Процессор — это «мозг» ЭВМ. Но для работы «мозга» нужна память, где хранились бы исходные данные и результаты вычислений. Эти функции выполняет оперативная память. Кроме данных, в оперативной памяти хранится и программа, которую выполняет процессор. Памятью ЭВМ называется совокупность устройств, служащих для запоминания, хранения и выдачи информации, необходимой процессору для решения данной задачи. Принцип хранимой в памяти программы в 1945 г. сформулировал американский математик Дж. фон Нейман. Совокупность процессора и оперативной памяти называют ядром ЭВМ. Для выполнения работы ЭВМ должна «знать», какую программу следует выполнить, какие использовать данные, куда выдать результат. Эти функции реализуются с помощью системы ввода-вывода. Система ввода-вывода обеспечивает обмен информацией между ядром ЭВМ и внешней средой. Внешние устройства ЭВМ, являющиеся компонентами системы ввода-вывода, преобразуют форму представления информации, используемую в ЭВМ, в форму, пригодную для восприятия объектом или оператором, и обратно. Через систему ввода-вывода к ядру ЭВМ подключаются внешние запоминающие устройства (внешняя память) — магнитные диски, ленты и т. п.

Современные ЭВМ выполняют свои функции на основе принципов программного управления и хранимой в памяти программы. При этом программа и данные хранятся в памяти ЭВМ. Оперативная память представляет собой совокупность пронумерованных ячеек для хранения команд и данных. Номер ячейки — это ее адрес. В командах указываются не сами числа, над которыми выполняются операции, а адреса ячеек памяти, где они находятся, и адрес ячейки, куда помещается результат операции. Обычно команды, которые необходимо выполнить процессору, находятся в последовательных ячейках памяти, расположение команд соответствует порядку их выполнения. Исключение составляют специальные команды, изменяющие последовательный порядок при выполнении определенных условий или безусловно.

Рассмотренные принципы функционирования являются общими для ЭВМ различных типов, однако для микро-ЭВМ есть определенные отличия в организации взаимодействия основных компонентов ЭВМ. Эти отличия связаны с тем, что в микро-ЭВМ составные части соединяются между собой

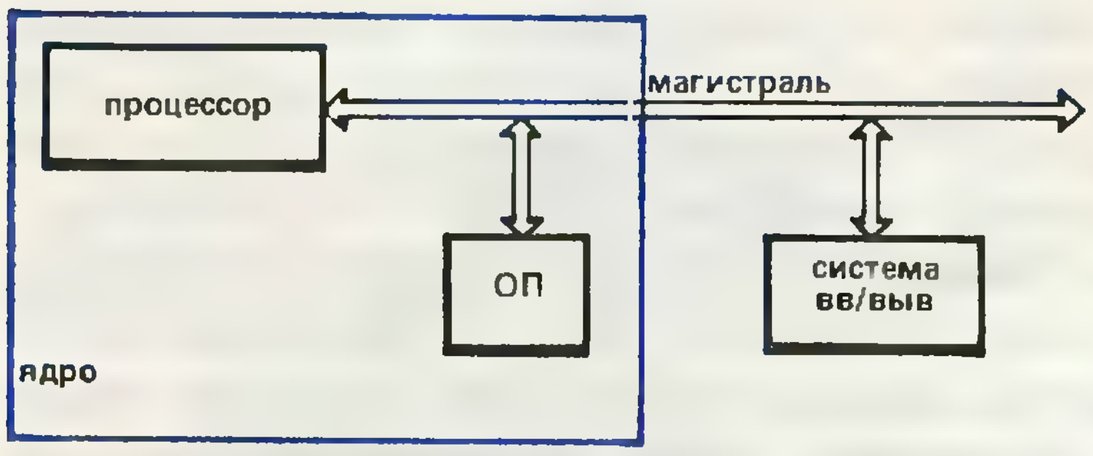


Рис. 1. Структура микро-ЭВМ

множеством сигнальных проводов, называемых шиной (магистралью). При этом структура микро-ЭВМ имеет вид, показанный на рис. 1. В более сложных ЭВМ используются несколько магистралей для связи между отдельными частями: процессор — память, процессор — система ввода-вывода и т. п.

Данные, которыми обмениваются процессор, память и система ввода-вывода, в микро-ЭВМ передаются по общей магистрали. Процессор извлекает из памяти команду, определяет, где находятся данные, с которыми надо работать, выполняет команду, переходит к следующей. Если необходимо, процессор через систему ввода-вывода может передать данные оператору или считать данные, которые вы хотите ввести. Изменяя содержимое памяти, мы можем изменять программу, выполняемую процессором, заменяя устройства, подключенные к системе ввода-вывода, мы можем заставить процессор управлять различными объектами. Но только надо иметь в виду, что процессор, конечно, может многое, но он не может «терпеть» компромиссов. Для процессора нет полутонов: либо черное, либо белое, только «да» или «нет», ноль или единица. Других цифр процессор не понимает, он работает с данными, представленными в двоичной системе счисления. Не думайте, что это так мало — различать всего две цифры, ведь машинное слово может содержать много цифр (обычно 4, 8, 16, 32 — для микропроцессоров). Эти вереницы нулей и единиц, трудно воспринимаемые человеком, для процессора родной язык, здесь и лежат все трудности нашего взаимопонимания. Для нас это нули и единицы, собранные в группы, а для процессора это и буквы, и цифры, и команды.

Представление информации в ЭВМ

Как мы уже отметили ранее, язык машинных команд — это язык нулей и единиц. Длина «слова» ЭВМ фиксирована и обычно кратна 8. Один двоичный разряд машинного слова называется «бит». Бит может принимать значение 0 или 1; 8 бит составляют байт. Слово — это одна группа обрабатываемых бит, единое выражение или одна команда микропроцессора. В зависимости от разрядности процессора слово может быть равно 8 битам, т. е. байту (для восьмиразрядного процессора), двум байтам (для 16-разрядного), четырем байтам (для 32-разрядного процессора). В одной ячейке памяти хранится обычно байт информации (минимальной адресуемой единицей информации является байт). Максимальное десятичное число, задаваемое восьмиразрядным двоичным числом, равно 255. То есть один байт может иметь 256 значений (от 0 до 255). Вся информация, хранимая в памяти ЭВМ, представляется двоичными числами, но тип этой информации может быть различным. Процессор различает следующие типы данных: число с фиксированной запятой, число с плавающей запятой, символьная информация, код команды. Пример того, как два байта информации могут быть поняты центральным процессором ЭВМ «Электроника-60», показан на рис. 2. Таким образом, один и тот же набор нулей и единиц процессор может рассматривать как код команды, число или код символа.

Тип информации, хранимой в ячейках памяти, определяется процессором в ходе выполнения программы. Представление информации в двоичном коде очень удобно процессору, но не удобно человеку, поэтому при записи команд и данных используется не двоичная, а восьмеричная или шестнадцатеричная система счисления.

Рис. 2. Пример преобразования двоичного числа в восьмеричное

Восьмеричная система содержит 8 цифр от 0 до 7, т. е. является системой с основанием восемь. Для записи одной цифры восьмеричного символа требуется три двоичных разряда. В табл. 1 представлено соответствие чисел в двоичной, восьмеричной и десятичной системах счисления. Для преобразования двоичного числа в его восьмеричный эквивалент необходимо выполнить следующие действия: нужно разделить двоичное, начиная с младших разрядов, на триады, затем, используя табл. 1, преобразовать каждую триаду в эквивалентную восьмеричную цифру. Пример преобразования двоичного числа из двух байт в восьмеричную форму показан на рис. 3.

Таблица 1

Десятичные	Восьмеричные	Двоичные		
		4	2	1
0	0	0	0	0
1	1	0	0	1
2	2	0	1	0
3	3	0	1	1
4	4	1	0	0
5	5	1	0	1
6	6	1	1	0
7	7	1	1	1

Запись числа в восьмеричной форме более компактная, однако к недостатку такой формы представления чисел относится несовпадение границы триады с границами байт и наличие неполных триад. Это приводит к тому, что, если двухбайтное слово, показанное на рис. 2, представить в виде двух раздельных байт, их восьмеричное представление будет не 134 и 645, а 271 и 245.

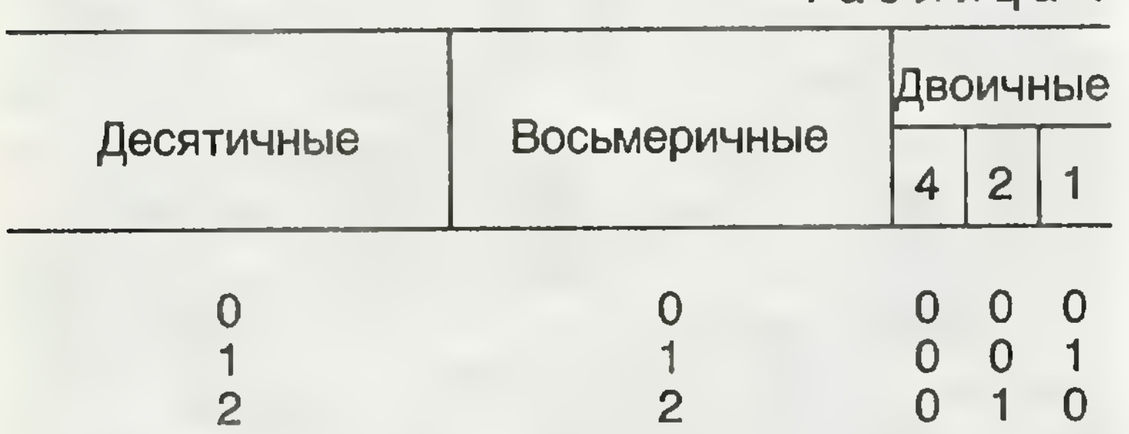


Рис. 3. Двоичное представление информации и ее интерпретация центральным процессором ЭВМ «Электроника-60»

Шестнадцатеричная система счисления. Для записи чисел в шестнадцатеричной системе счисления используются цифры от 0 до 9 и буквы A, B, C, D, E, F. Один символ шестнадцатеричной системы счисления может быть представлен тетрадой (четыре бита) в двоичной системе счисления. Соответствие между десятичным, шестнадцатеричным и двоичным представлением чисел приведено в табл. 2.

Таблица 2

Десятичные	Шестнадцатеричные	Двоичные			
		8	4	2	1
0	0	0	0	0	0
1	1	0	0	0	1
2	2	0	0	1	0
3	3	0	0	1	1
4	4	0	1	0	0
5	5	0	1	0	1
6	6	0	1	1	0
7	7	0	1	1	1
8	8	1	0	0	0
9	9	1	0	0	1
10	A	1	0	1	0
11	B	1	0	1	1
12	C	1	1	0	0
13	D	1	1	0	1
14	E	1	1	1	0
15	F	1	1	1	1

Для представления двоичного числа в шестнадцатеричной форме следует разбить число на тетрады, начиная с младших разрядов, затем каждую тетраду заменить соответствующим шестнадцатеричным эквивалентом, используя табл. 2. Пример представления двухбайтового числа в шестнадцатеричной системе счисления показан на рис. 4. Из примера видно, что границы тетрад и байтов совпадают, поэтому код двухбайтного числа состоит из шестнадцатеричных кодов байтов, составляющих это число.

Использование шестнадцатеричной

Рис. 4. Пример преобразования двоичного числа в шестнадцатеричное

и восьмеричной систем счисления позволяет более компактно представлять двоичную информацию, хранимую в памяти ЭВМ, и адреса ячеек памяти. Данные, хранимые в памяти и обрабатываемые процессором, делятся на три группы: логические коды, числа с фиксированной запятой, числа с плавающей запятой.

Логическими кодами могут быть представлены целые числа в диапазоне от 0 до 255 для байта и от 0 до 65 534 для слова; битовые величины — задающие значения отдельных разрядов байта или слова; символьные величины. Эти коды могут размещаться в отдельных байтах или словах.

Символьные величины задаются в коде КОИ-7. Для представления символа отведен 1 байт, причем используются только семь младших разрядов. Код символа представляется либо в восьмеричной (для микро-ЭВМ «Электроника-60»), либо в шестнадцатеричной (для микропроцессора К 580ИК80 и других) системах счисления (табл. 3).

Таблица 3

Символ	Восьмеричная система счисления	Шестнадцатеричная система счисления	Двоичный код
!	41	21	00100001
*	52	2A	00101010
/	57	2F	00101111
0	60	30	00110000
7	67	37	00110111
A	101	41	01000001
D	104	44	01000100

Помимо печатных символов, кодируются некоторые управляющие коды клавиатуры терминала. Например: перевод строки (ПС) — 12_8, $\varnothing A_{16}$; возврат каретки (ВК) — 15_8, $\varnothing Д_{16}$; звонок — 7_8, 7_{16}.

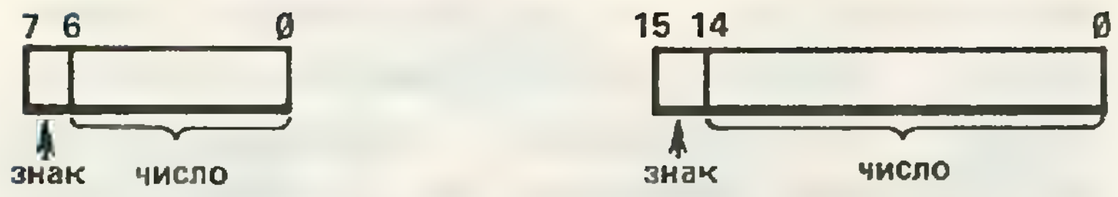


Рис. 5. Представление чисел с фиксированной запятой

Числа с фиксированной запятой могут занимать слово или байт. Если бит знака равен 1 — число отрицательное, а если равен 0 — число положительное. Пример представления чисел с фиксированной запятой показан на рис. 5. Диапазон представления чисел — $-128_{10} \div +127_{10}$ — байт; — $32768_{10} \div +32767_{10}$ — слово. Представление чисел с плавающей запятой имеет более сложную форму.

Структура языка АССЕМБЛЕР

АССЕМБЛЕР занимает промежуточное место между языками высокого уровня и машинными кодами. Язык АССЕМБЛЕР является машинно-зависимым и отражает аппаратурные особенности той микро-ЭВМ, для которой он был разработан.

Близость к микро-ЭВМ определяется тем, что в языке АССЕМБЛЕР используются мнемонические (буквенные) обозначения машинных команд, учитывается ее внутренняя структура и формы представления чисел. Вместе с тем есть общие черты у АССЕМБЛЕРов различных микро-ЭВМ.

Программа, написанная на АССЕМБЛЕРе, состоит из последовательности операторов (причем каждый оператор соответствует обычно одной машинной команде). В этом языке обычно выделяют следующие группы операторов: операторы машинных команд, операторы псевдокоманд, команды управления ассемблером (программой).

Операторы машинных команд соответствуют символической записи этих команд микро-ЭВМ. Операторы псевдокоманд позволяют описывать данные, резервировать ячейки памяти, указывать начало и конец программ-

ного модуля и т. п. Команды управления ассемблером дают возможность пользователю «командовать» самим ассемблером, т. е. указывать, как именно должна быть обработана программа, какую дополнительную информацию должен выдавать ассемблер в процессе своей работы. Команды этого типа, вообще говоря, не являются частью языка АССЕМБЛЕР, так как используются только в процессе работы программы ассемблера, т. е. на этапе трансляции.

Обычно оператор языка АССЕМБЛЕР представляет собой строку символов и цифр и содержит четыре поля:

Поле метки	Поле операции	Поле операндов	Комментарий

Все поля разделяются специальными знаками, позволяющими ассемблеру при «расшифровке» оператора точно определить границы полей.

Поле метки содержит символическое имя оператора, которое может состоять из последовательности букв и цифр, максимальная длина метки обычно не превышает восьми знаков.

Поле операции содержит мнемоническую запись операции, выполняемой оператором. В поле операции могут быть указаны в мнемоническом виде коды машинных команд, псевдокоманд, команд управления ассемблером.

В поле операндов указывается один или несколько операндов, с которыми следует выполнять операцию. Операнды могут быть числами, символическими именами и выражениями.

В поле комментария может быть помещен любой текст, который облегчает программисту чтение своей программы. Комментарий никак не влияет на выполнение оператора.

В операторе не обязательно должны быть заполнены все поля, некоторые операции не требуют операндов, метки ставятся обычно только у того оператора, к которому возможно обращение в каком-либо другом операторе программы. Метка отделяется от поля

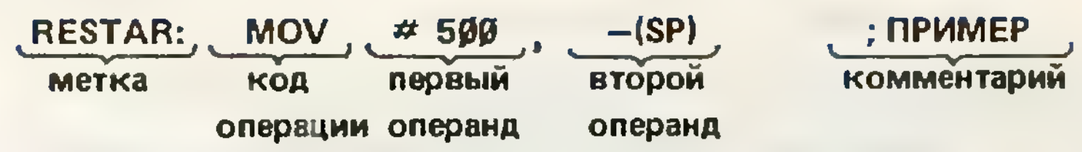


Рис. 6. Оператор языка АССЕМБЛЕРа микро-ЭВМ «Электроника-60»

операции обычно двоеточием, поле операции от поля операндов пробелом, операнды отделяются друг от друга запятыми, а поле комментария начинается с точки с запятой. Размеры полей, мнемоника команд, способы задания операндов и разделители полей различны для различных ассемблеров. На рис. 6 показан пример оператора, написанного на АССЕМБЛЕРе микро-ЭВМ «Электроника-60».

Программные модели микропроцессорных систем

АССЕМБЛЕР ориентирован на определенный тип ЭВМ, поэтому при изучении этого языка недостаточно знать общую структурную схему микро-ЭВМ. Необходимо уточнить ее и перейти к так называемой программной (регистровой) модели микро-ЭВМ. Такая модель включает только те элементы ЭВМ, к которым можно обращаться программно на языке АССЕМБЛЕРа.

Основным элементом программной модели является регистр, т. е. некоторое физическое устройство, способное воспринимать, хранить и выдавать информацию. Разрядность всех регистров равна или кратна длине слова микропроцессора. Память ЭВМ можно рассматривать как совокупность регистров, имеющих адреса. Функционирование микро-ЭВМ (выполнение программы) можно рассматривать как передачу данных между регистрами процессора, памяти и устройств ввода-вывода. Программная модель микропроцессорной системы, построенной на основе микропроцессорного комплекта К 580, показана на рис. 7.

В программную модель входят внутренние регистры микропроцессора, находящиеся внутри самого МП, и внешние регистры, т. е. представляющие оперативную память и систему ввода-вывода.

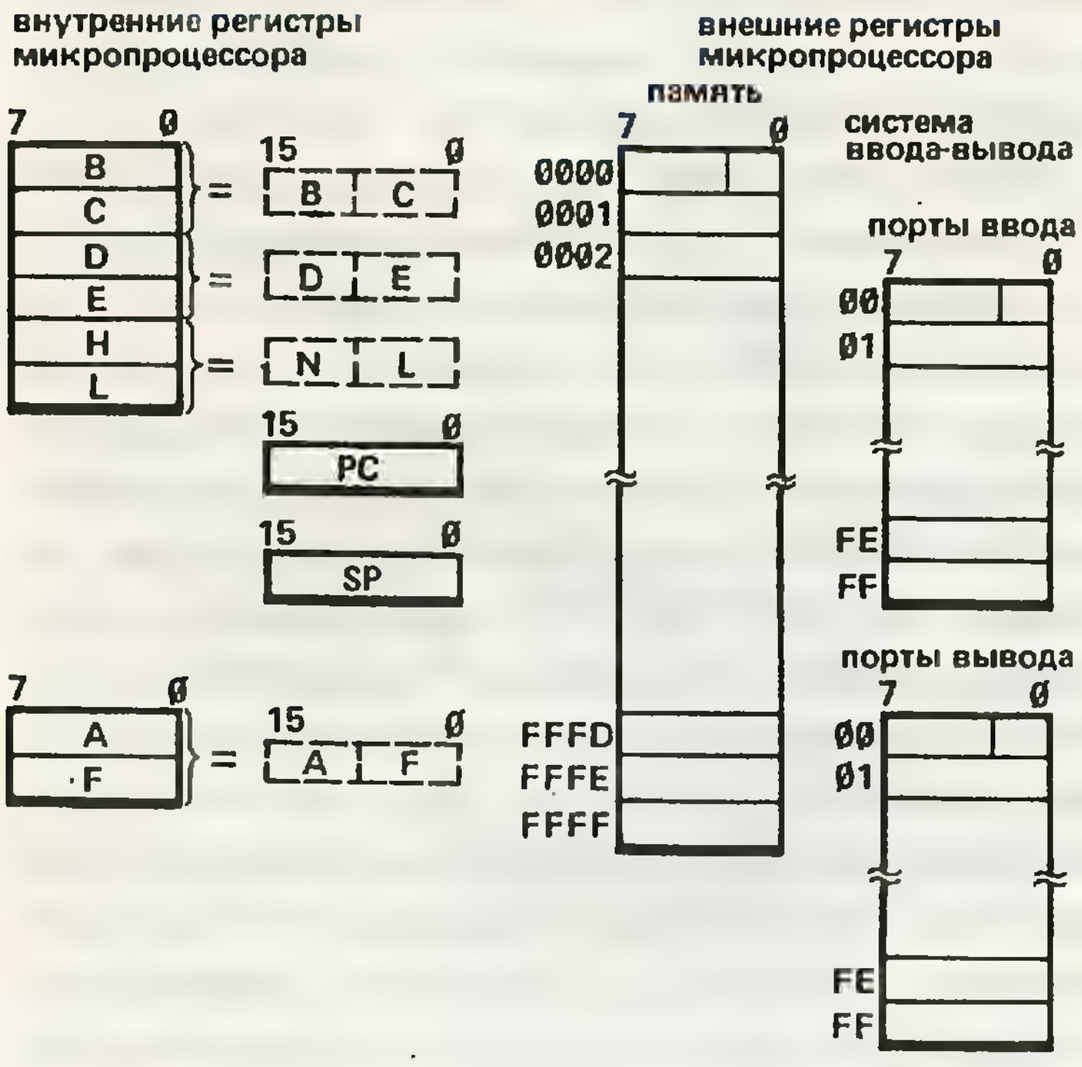


Рис. 7. Программная модель МП-системы на основе МП К 580

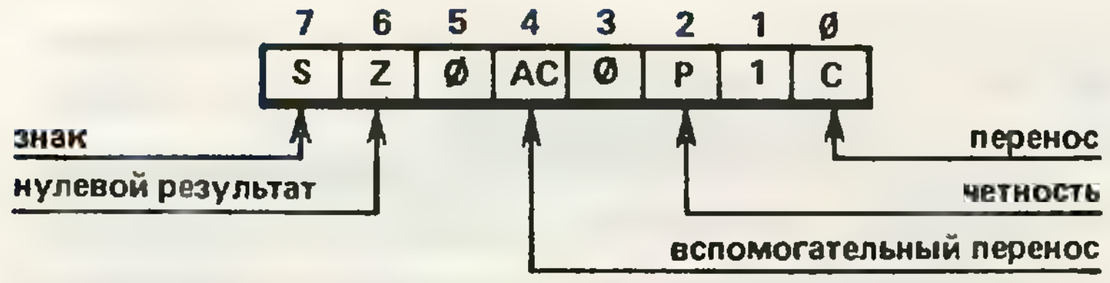


Рис. 8. Формат регистра признаков F.

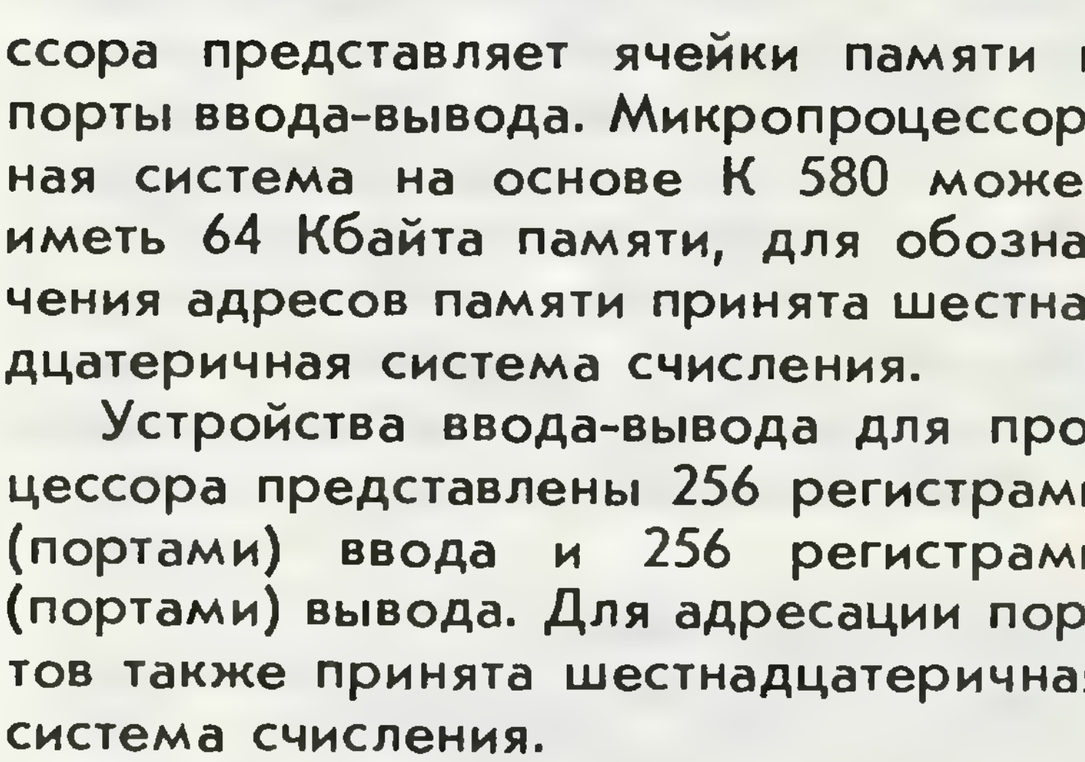


Рис. 9. Программная модель микро-ЭВМ «Электроника-60»

Регистр А — аккумулятор, является основой для выполнения всех операций МП, к которым относятся арифметические и логические, операции загрузки, размещения данных и ввода-вывода.

Регистры общего назначения B, C, D, E, H, L могут быть использованы как шесть 8-разрядных регистров или объединены в три 16-разрядные регистровые пары в зависимости от текущей выполняемой команды.

Программный счетчик (PC) — шестнадцатиразрядный регистр — указывает на ячейку памяти, где расположена команда, выполнение которой начнется вслед за текущей.

Указатель стека (SP) — шестнадцатиразрядный регистр, указывает на начальный адрес особой области памяти для временного хранения данных, такая область называется стеком.

F — регистр признаков, содержит пять одноразрядных индикаторов, в которых имеется информация, определяющая состояние МП. Формат регистра F показан на рис. 8.

Содержимое регистров A и F определяют текущее состояние МП, поэтому пару регистров A, F, нередко называют регистром состояния микропроцессора.

«Внешний» мир для микропроцессора представляет ячейки памяти и порты ввода-вывода. Микропроцессорная система на основе К 580 может иметь 64 Кбайта памяти, для обозначения адресов памяти принята шестнадцатеричная система счисления.

Устройства ввода-вывода для процессора представлены 256 регистрами (портами) ввода и 256 регистрами (портами) вывода. Для адресации портов также принята шестнадцатеричная система счисления.

Программная модель микро-ЭВМ «Электроника-60» показана на рис. 9.

К внутренним регистрам процессора микро-ЭВМ относятся: R∅÷R7 — регистры общего назначения (16-разрядные). Все регистры можно использовать произвольно, однако традиционно R6 и R7 применяют как программный счетчик и указатель стека. Обычно: R7 — программный счетчик, используется для адресации (указания) следующей выполняемой команды.

R6 — указатель стека, содержит адрес вершины стека. R∅—R5 — используются по усмотрению программиста. РСП — 16-разрядный регистр состояния процессора, служит для управления процессором со стороны про-

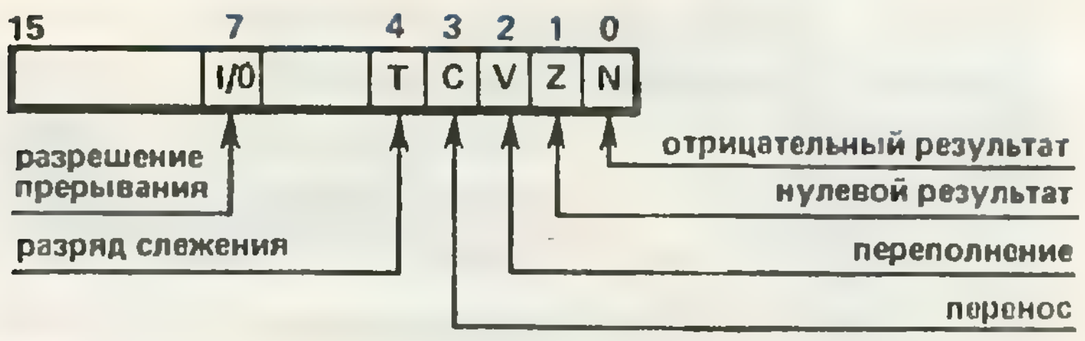


Рис. 10. Формат регистра состояния процессора микро-ЭВМ «Электроника-60»

граммы и для отображения его состояния, в частности признаков результата предыдущей операции. Формат регистра состояния процессора микро-ЭВМ «Электроника-60» показан на рис. 10.

Разряды РСП могут быть установлены программно специальными командами. В микро-ЭВМ «Электроника-60» для нумерации адресов ячеек памяти принята восьмеричная система счисления. Память составляет 56 Кбайт с адресами от 0_8 до 157777_8. Микро-ЭВМ может работать с байтами и со словами, поэтому два смежных байта образуют слово. При этом адрес слова всегда четное число и совпадает с адресом младшего байта. Для подключения устройств ввода-вывода к микро-ЭВМ «Электроника-60» используется область памяти объемом 8 Кбайт с адресами от 160000_8 до 177777_8. Таким образом, работа процессора с внешними устройствами и ячейками памяти не имеет принципиальных отличий.

Программирование микропроцессорных систем на языке АССЕМБЛЕРа

Операторы машинных команд АССЕМБЛЕРа соответствуют мнемоническому представлению машинных команд микропроцессора. Состав машинных команд определяется типом, аппаратной структурой, функциональными возможностями микропроцессора. Обычно все команды можно сгруппировать следующим образом: команды передачи данных, логических и арифметических операций, команды передачи управления. Рассмотрим операторы машинных команд АССЕМБЛЕРа на примерах микропроцессора КР580ИК80 и микро-ЭВМ «Электроника-60», которые широко используются на практике. При описании примеров мы будем обращаться к программной модели соответствующей микропроцессорной системы.

Как отмечалось ранее, функционирование ЭВМ можно рассматривать как передачу информации между регистрами МП, памятью и системой ввода-вывода. Непосредственно передачу данных выполняют команды пересылки. В вычислительной системе на базе МП КР580ИК80 возможны следующие пересылки: 1) из внутреннего регистра микропроцессора во внутренний регистр; 2) из внутреннего регистра в память или порт; 3) из памяти или порта во внутренний регистр. Каждая команда передачи содержит адрес источника и получателя данных.

Примеры: 1) Переслать содержимое регистра B (который содержит число 27_{16}) в регистр C (содержащий число $\varnothing A5_{16}$) (смотри программную модель МП-системы на базе МП КР580ИК80). Команда, выполняющая это действие, имеет в памяти ЭВМ числовой код 48_{16}, а на языке АССЕМБЛЕРа мнемоническое обозначение: MOV C, B. После выполнения команды регистр C будет содержать число 27_{16}, а регистр B своего содержимого не изменит.

2) Переслать содержимое регистровой пары HL($\varnothing F\varnothing A2_{16}$) в регистр указатель стека SP, который содержит число 5723_{16}. Команда имеет числовой код $\varnothing F9_{16}$ и мнемоническое обозначение: SPHL. После выполнения команды регистр-указатель стека SP будет содержать новое значение $\varnothing F\varnothing A2_{16}$, а содержимое регистровой пары HL не изменится.

3) Записать содержимое регистра A ($\varnothing FF_{16}$) в ячейку памяти с адресом $\varnothing 5F\varnothing_{16}$. Код данной команды $32F\varnothing\varnothing 5_{16}$ занимает в памяти ЭВМ 3 байта, а мнемоника на языке АССЕМБЛЕРа имеет вид: STA $\varnothing$5F$\varnothing$H. Выполнение команды приведет к тому, что в ячейку памяти с адресом $\varnothing 5F\varnothing_{16}$ будет записано число FF_{16}.

4) Ввести в регистр A содержимое порта 05_{16}. Код данной команды $\varnothing DB\varnothing 5_{16}$ занимает 2 байта, а мнемоника имеет вид: IN 05H.

Содержимое 5 порта будет введено в регистр А. Надо отметить, что в МП-системе на основе МП КР580ИК80 все операции обращения к портам ввода-вывода — через регистр А.

В микро-ЭВМ «Электроника-60», помимо указанных команд, имеются команды, выполняющие обмен между ячейками памяти, а внешние устройства представлены наборами регистров, адресуемых как ячейки памяти с фиксированными адресами. Надо отметить, что в системе команд МП КР580ИК80 уже подразумевается способ адресации операндов, в то время как в командах микро-ЭВМ «Электроника-60» метод адресации выбирается программистом.

Примеры: 1) Переслать содержимое регистра R1 ($\varnothing 12756_8$) в регистр R2. Код команды $\varnothing 1\varnothing 1\varnothing 2_8$, а мнемоника имеет вид: MOV R1, R2. Содержимое регистра R2 после выполнения команды будет $\varnothing 12756_8$. (Смотри программную модель микро-ЭВМ «Электроника-60».)

2) Переслать содержимое регистра R3 (177562_8) в ячейку памяти с адресом 1030_8. Код команды[1] будет занимать два слова (т. е. 4 байта): 010337_8 001030_8, а мнемоника имеет вид: MOV R3, ⊙≠1∅3∅. В ячейку памяти <∅3∅$_8$ запишется 177562_8.

3) Записать в регистр R4 число 75_8. Код команды занимает два слова $\varnothing 127\varnothing 4_8$, $\varnothing\varnothing\varnothing\varnothing 75_8$, ее мнемоника имеет вид: MOV≠75, R4.

4) Переписать содержимое ячейки памяти с адресом $1\varnothing\varnothing\varnothing_8$ (177552_8) в ячейку памяти с адресом $2\varnothing 2\varnothing_8$. Код команды содержит три слова: $\varnothing 13737_8$, $\varnothing\varnothing 1\varnothing\varnothing\varnothing_8$, $\varnothing\varnothing 2\varnothing 2\varnothing_8$, а мнемоника имеет вид: MOV ⊙≠ 1∅∅∅, ⊙2∅2∅.

Как видно из примеров, одна и та же команда MOV-пересылки осуществляет самые разнообразные передачи данных, так как в каждом конкретном операторе используется свой способ адресации источника и приемника информации. В микро-ЭВМ «Электроника-60» используется двенадцать способов адресации. Знание всех способов адресации и умелое их использование позволяют создавать компактные программы, занимающие минимальный объем памяти.

Группа команд логических и арифметических операций. Помимо пересылки информации, ЭВМ выполняет вычислительные процедуры. На АССЕМБЛЕРе это выполняется с помощью команд арифметических и логических операций. Разные ЭВМ обладают разными вычислительными возможностями. Например, в МП КР580ИК80: арифметические операции ограничены действиями: 1) Сложение; 2) Вычитание; 3) Инкремент (увеличение на 1); 4) Декремент (уменьшение на 1); 5) Сравнение аккумулятора.

Примеры: 1) Сложить содержимое регистра L ($\varnothing 5_{16}$) с содержимым аккумулятора А ($2\varnothing_{16}$). Код команды 85_{16}, а мнемоника: ADD L. После выполнения команды регистр А будет содержать 25_{16}. (Смотри программную модель МП-системы на базе МП КР580ИК80.)

2) Вычесть из содержимого аккумулятора А ($3\varnothing_{16}$) число 4. Код команды содержит 2 байта: $\varnothing 6_{16}$, $\varnothing 4_{16}$, а мнемоника: SUJ 4H. После выполнения команды регистр А будет содержать $2C_{16}$.

3) Увеличить на 1 содержимое регистра С ($\varnothing A_{16}$). Код команды $\varnothing C_{16}$, а мнемоника: INR С. После выполнения в регистре С будет находиться число $\varnothing B_{16}$.

4) Уменьшить на 1 содержимое регистровой пары DE ($\varnothing A6\varnothing_{16}$). Код команды $1B_{16}$, а мнемоника: DCX D. После выполнения команды регистровая пара содержит число $\varnothing A5F_{16}$.

5) Сравнить содержимое регистра А ($\varnothing B_{16}$) с содержимым регистра С ($\varnothing B_{16}$). Код команды $\varnothing B9_{16}$, а мнемоника: CMP С. Результатом данной команды являются выставленные индикаторы в регистре флагов F. (Смотри программную модель МП-системы на МП КР580ИК80.) Так как содержимое регистра совпадает, будет установлен в единицу флаг Z (разряд 6).

---

[1] Так как в наборе нет знака «коммерческая эта», вместо него будем использовать ⊙, а вместо знака «диез» — ≠.

Надо отметить, что одним из регистров в операциях сложения и вычитания в МП КР580ИК80 всегда является регистр А — аккумулятор.

В МП КР580ИК80, помимо арифметических, существует ряд команд логических операций: И, ИЛИ, исключающее ИЛИ, инверсия и сдвиг. Операции выполняются побитно над содержимым аккумулятора и второго операнда, который указывается в мнемонике команды. Результат всегда хранится в аккумуляторе.

Примеры: 1) Выполнить логическую операцию ИЛИ над содержимым аккумулятора А ($8\varnothing_{16}$) и регистра L ($\varnothing F_{16}$). Код команды $\varnothing B5_{16}$, а мнемоника: ORA L.

После выполнения команды в аккумуляторе будет число $8F_{16}$.

2) Выполнить логическую операцию И над содержимым аккумулятора А (13_{16}) и числом 2. Код команды занимает 2 байта $\varnothing E6\ \varnothing 2_{16}$, а мнемоника: ANI $\varnothing$2H. Результатом будет содержимое регистра А равное 2.

3) Выполнить операцию исключающее ИЛИ над содержимым аккумулятора ($\varnothing A5_{16}$). Код команды $\varnothing AF_{16}$, а мнемоника: XRA A. Результатом будет обнуленный аккумулятор А и выставленный флаг нуля Z (разряд 6) в регистр флагов F.

4) Инвертировать содержимое аккумулятора А ($\varnothing F_{16}$). Код команды $2F_{16}$, а мнемоника: CMA. Результат: в аккумуляторе число $\varnothing F\varnothing_{16}$.

5) Осуществить сдвиг вправо на один разряд содержимого аккумулятора А ($2A_{16}$), мнемоника: RAR. Результат: в аккумуляторе число 15_{16}.

В более мощной микро-ЭВМ «Электроника-60», помимо приведенных команд, существуют другие. Например, умножение, деление, сложение и вычитание чисел с плавающей запятой. Эти команды позволяют микро-ЭВМ эффективно выполнять сложные вычислительные процедуры. Большинство арифметических и логических команд, а также некоторые другие влияют на установку (или сброс) признаков (индикаторов) в регистре флагов F. Содержимое регистра F может анализироваться командами передачи управления. Управление передается в зависимости от того, выставлен в 1 или сброшен в 0 тот или иной признак или группа признаков. Эти команды микропроцессора позволяют создавать программы, которые способны адаптироваться к сладывающейся ситуации, анализировать, принимать решение.

Пример 1) Передать управление на метку M1, в случае если выставлен признак нуля. На АССЕМБЛЕРе МП КР580ИК80 эта команда имеет мнемонику: JZ M1. На АССЕМБЛЕРе микро-ЭВМ «Электроника-60» мнемоника иная: BEQ M1. Помимо передачи управления, по условию существуют команды безусловной передачи управления, вызова подпрограмм, возврата из подпрограммы и т. д.

Описанные группы команд АССЕМБЛЕРа не исчерпывают полного набора команд конкретного микропроцессора, но включают в себя наиболее распространенные машинные команды. Для удобства пользователя в АССЕМБЛЕРе существуют операторы псевдокоманд и команды управления ассемблером. Эти операторы выполняются на этапе трансляции, т. е. перевода исходной программы в машинные коды.

Набор псевдокоманд определяется используемой программой ассемблер. На примере АССЕМБЛЕРов микро-ЭВМ «Электроника-60» и СМ-1800 (построенной на базе МП КР580ИК80) рассмотрим наиболее часто встречающиеся операторы псевдокоманд:

1) Присвоение. Присвоить переменной VALUE значение $127_{1\varnothing}$

а) СМ-1800: VALUE EQU 127
б) Электроника-60: VALUE=127.

2) Резервирование памяти. Зарезервировать память объемом $1\varnothing_{1\varnothing}$ байт, начиная с адреса REZ:

а) СМ-1800: REZ: DS 1∅
б) Электроника-60: REZ: .BLKB 1∅.

3) Определение данных. Задать значение байта KOL равное 5:
а) СМ-1800: KOL: DB 5
б) Электроника-60: KOL: ⊙BYTE 5

4) Определение данных. Задать значение слова (2 байта) TIME равное 272_{10}
а) СМ-1800: TIME: DW 272
б) Электроника-60: TIME: ⊙WORD 272.

5) Конец программы (исходного модуля)
а) СМ-1800:
б) Электроника-60: .END

В АССЕМБЛЕРе микро-ЭВМ «Электроника-60» набор псевдокоманд более широкий, чем в АССЕМБЛЕРе микро-ЭВМ СМ-1800.

Операторы псевдокоманд позволяют задавать макрокоманду. Макрокомандой называется такой оператор в языке АССЕМБЛЕРа, который при трансляции заменяется последовательностью других операторов языка. Эту последовательность, в свою очередь, называют макрорасширением макрокоманды. Каждой макрокоманде соответствует, кроме того, макроопределение, которое может находиться в том же программном модуле, что и макрокоманда, или чаще всего в библиотеке макроопределений. Макроопределение представляет собой совокупность операторов, используемых ассемблером для формирования макрорасширения всякий раз, когда в исходном модуле встречается макрокоманда. В целом макроопределение можно понимать как программную «заготовку» для получения макрорасширения, причем в общем случае размер и характер макрорасширения зависит от параметров, записанных в макрокоманде. Совокупность макрокоманд и средств их обработки в ассемблере называют макросредствами языка АССЕМБЛЕР.

Управление внешним устройством от микропроцессора

Любое внешнее устройство может найти «общий язык» с микропроцессором, если информация, получаемая и выдаваемая этим устройством, представлена в двоичном коде. «Общение» микропроцессора и внешнего устройства состоит в обмене битами, байтами или словами. При этом способ взаимодействия определяется типом внешнего устройства и алгоритмом его работы.

В качестве примера рассмотрим светофор, управляемый микропроцессорной системой. Моменты переключения светофора определяет микропроцессор. Алгоритм работы светофора представлен в таблице:

Номер	Дорога 1	Дорога 2	Время, с
1	Зеленый	Красный	30
2	Желтый	Красный и желтый	5
3	Красный	Зеленый	15
4	Красный и желтый	Желтый	5

Управление светофором осуществляется через регистр управления. Формат управляющего слова показан на рис. 11.

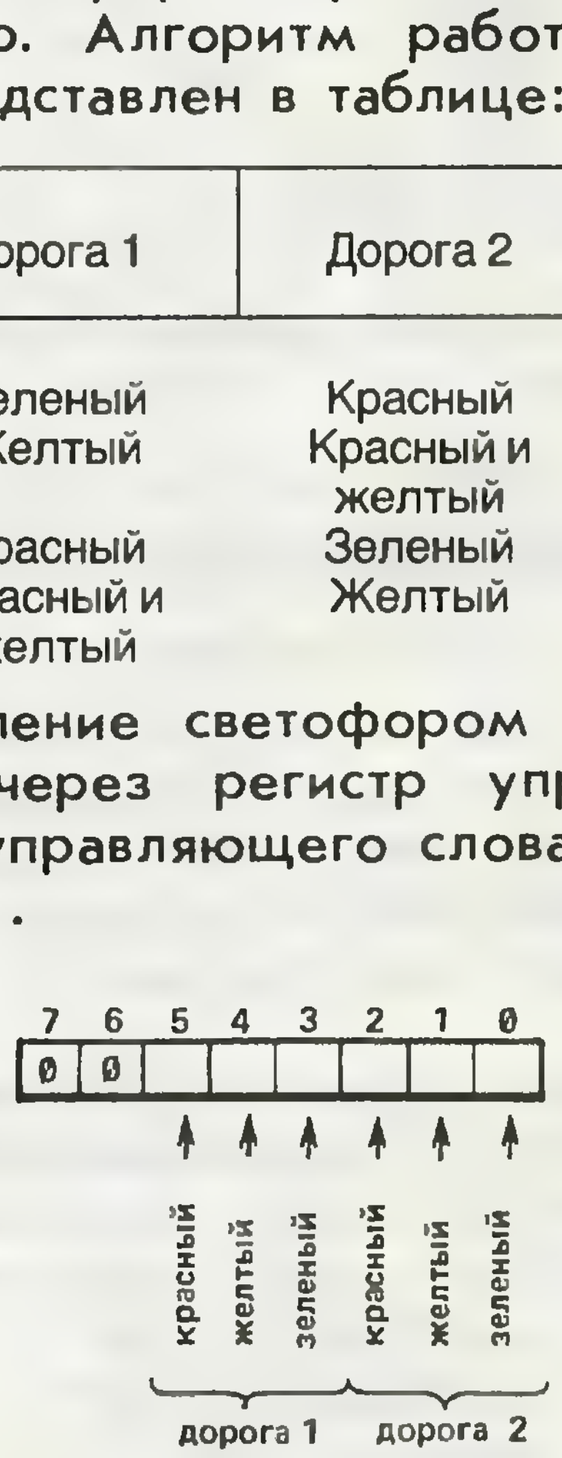


Рис. 11. Формат регистра управления светофором

При единичном значении бита соответствующий сигнал зажигается, при нулевом — гаснет.

Для отсчета временных интервалов предусмотрен таймер. Запуск таймера осуществляется подачей на его вход временного интервала в секундах. После окончания временного интервала на выходе таймера появляется сигнал.

Осуществим управление светофором с помощью микропроцессорной системы на основе МП КР580ИК80. Подключим регистр управления к нулевому выходному каналу, регистр таймера к первому выходному каналу, а сигнал таймера будет считываться по нулевому входному каналу. Программа управления светофором будет иметь вид:

```
; программа управления светофором

UPR     EQU     Ø       ; канал подачи управляющего слова
RKST    EQU     Ø       ; канал приема сигнала с таймера
RDT     EQU     1       ; канал запуска таймера
LOOP:   LXI     B, COL
        LXI     D, TIME
        MVI     H, 4    ; загрузка счетчика цикла
M1:     LDAX    B       ; загрузка в А управляющего слова
        OUT     UPR     ; выдача управляющего слова
        LDAX    D       ; загрузка в А значения временного интервала
        OUT     RDT     ; запуск таймера
M2:     IN      RKST    ; ввод сигнала с выхода таймера
        ANA     A       ; установка признака нуля
        JZ      M2      ; переход на повторный ввод при нуле
        INX     B
        INX     D
        DCR     H       ; уменьшение содержимого счетчика цикла
        JNZ     M1
        JMP     LOOP    ; переход на начало программы
TIME:   DB 3Ø,5,15,5    ; временные интервалы для четырех состояний
COL:    DB 14Q, 26Q 41Q, 62Q    ; управляющие слова для четырех состояний
        END
```

Выполнение программы прерывается только при выключении питания.

Решим поставленную задачу с использованием вычислительной системы на базе микро-ЭВМ «Электроника-60». Для этого подключим регистр управления светофором к адресу $1637ØØ_8$, регистр таймера к адресу $1637Ø2_8$, а сигнал управления к седьмому разряду регистра с адресом $1637Ø4_8$. Программа на АССЕМБЛЕРе будет иметь следующий вид:

```
; программа управления светофором
UPR=1637ØØ  ; адрес регистра управления светофором
RDT=1637Ø2  ; адрес регистра таймера
RKST=1637Ø4 ; адрес регистра приема сигнала с таймера.
LOOP:   MOV#COL, R1
        MOV#TIME, R2
        MOV#4, R3           ; загрузка счетчика цикла
M1:     MOVB(R1), UPR       ; выдача управляющего слова
        MOVB (R2), RDT      ; запуск таймера
```

```
M2:     TSTB RKST                ; опрос сигнала таймера
        BPL     M2               ; сигнала нет, повторить опрос
        INCB    R1
        INCB    R2
        DEC     R3               ; уменьшение содержимого
                                   цикла
        BNE      M1
        JMP      LOOP          ; переход на начало программы
TIME:  .BYTE 3∅., 5., 15., 5.  ; временные интервалы для четырех со-
                                 стояний
COL:    .BYTE 14, 26, 41, 62   ; управляющие слова для четырех со-
                                 стояний
        .END     LOOP
```

Выполнение программы прерывается только при выключении питания.

Изменяя содержимое четырех ячеек памяти, начиная с ячейки TIME, можно легко менять время горения каждого цвета, а содержимое ячейки COL и трех последующих определяет порядок смены цветов и также может быть изменено без переделки всей программы.

Заключение

Вы ознакомились с основами программирования на языке АССЕМБЛЕР. Возможно, вы несколько разочарованы, ведь все не так просто, но мы не обещали вам легкой жизни. Зато теперь, проявив терпение и настойчивость, вы знаете, что происходит в «голове» компьютера, выполняющего ваши программы. Знание АССЕМБЛЕРа поможет вам в том случае, если вы программируете на ФОРТРАНе и ПАСКАЛе: отдельные подпрограммы вы можете писать на АССЕМБЛЕРе и использовать в основной программе на языке высокого уровня. Таким образом, можно объединить достоинства языков различных типов.

АССЕМБЛЕР незаменим при создании управляющих программ для МП-контроллеров, работающих в реальном масштабе времени. Достоинства языка АССЕМБЛЕР и его возможности, наверное, послужат вам хорошей компенсацией за те усилия, которые необходимо приложить для его освоения.

Литература

Архитектура и проектирование микроЭВМ. Организация вычислительных процессов / Под ред. Л. Н. Преснухина. — Кн. 1. — М.: Высшая школа, 1986.

Вигдорчик Г. В., Воробьев А. Ю., Проченко В. Д. Основы программирования на ассемблере для СМ ЭВМ. — М.: Финансы и статистика, 1983.

Балашов Е. П., Пузанков Д. В. Микропроцессоры и микропроцессорные системы. — М.: Радио и связь, 1981.

Толстых Б. Л., Талов И. Л. и др. Мини- и микро-ЭВМ семейства «Электроника». — М.: Радио и связь, 1987.

Алексенко А. Г., Галицын А. А., Иванников А. Д. Проектирование радиоэлектронной аппаратуры на микропроцессорах. — М.: Радио и связь, 1984.

Гилл А. Программирование на языке Ассемблера для PDP-11. — М.: Радио и связь, 1983.

Шнайдер А. Язык ассемблера для персонального компьютера фирмы IBM / Пер. с англ. — М.: Мир, 1988. — 406 с.

В. Л. ГОРБУНОВ

ОДНОКРИСТАЛЬНЫЕ МИКРО-ЭВМ

По габаритам современные ЭВМ в 300 тыс. раз меньше ЭВМ первого поколения, выполняют операции в 10 тыс. раз быстрее, потребляют значительно меньше энергии, более надежны и относительно дешевы. Стоимость одной условной операции, проводимой ЭВМ, снизилась за последние 25 лет в 100 тыс. раз.

Своими успехами вычислительная техника обязана микроэлектронике, которая, используя последние достижения науки, позволяет в настоящее время на площади полупроводника меньше одного квадратного сантиметра изготовить большую интегральную схему (БИС), содержащую сотни тысяч транзисторов. Развитие микроэлектроники повлияло и на структуру проектируемых вычислительных устройств. Обычно БИС изготавливаются в виде конструктивно оформленного изделия, имеющего чаще всего не более 100 внешних контактов. Необходимость ограничения числа внешних контактов при сохранении функциональных возможностей изделия заставили разработчиков искать при проектировании новые архитектурные решения. Одно из таких решений — микропроцессорные БИС, содержащие законченные функциональные блоки ЭВМ со всеми внутренними связями, имеющие минимальное число внешних сигналов. На микропроцессорных БИС можно собирать ЭВМ с различными характеристиками.

Области применения вычислительных машин разнообразны, и в каждом случае применения к ним предъявляются свои особые требования. Например, для выполнения научных расчетов требуются ЭВМ с большой точностью вычисления, для управления быстропротекающими процессами требуются быстродействующие ЭВМ, а для управления бытовыми приборами чаще всего достаточно относительно медленных, с малой разрядностью, но обязательно дешевых микро-ЭВМ.

Для перекрытия различных диапазонов применений микропроцессорные БИС проектируются комплектами. Среди этих комплектов можно выделить секционные микропроцессорные БИС, однокристальные микропроцессоры и однокристальные микро-ЭВМ.

Секционные микропроцессорные БИС представляют собой совокупность отдельных блоков, из которых можно формировать ЭВМ необходимой разрядности с нужным количеством внешних магистралей.

В однокристальных микропроцессорах в одной БИС содержится процессор фиксированной разрядности с фиксированной системой команд. Для построения микро-ЭВМ к микропроцессорной БИС подключают схемы, формирующие магистрали, обеспечивающие обмен информации между блоками, различные схемы памяти и электронные устройства (все это — так называемые БИС обрамления).

Однокристальные микро-ЭВМ (ОМЭВМ) содержат в одном кристалле все необходимые блоки для выполнения вычислений и согласованной работы блоков и устройств: процессор; оперативную память; память для записи и хранения рабочей программы; схемы, формирующие внутренние и внешние магистрали, и т. д. Как видим, структура ОМЭВМ достаточно сложна, но реализована она должна быть на ограниченной площади полупроводника и при жестком лимите по количеству структурных элементов. Для того чтобы разместились все необходимые блоки в одном кристалле, приходится ограничить разрядность, которая у однокристальных микро-ЭВМ обычно не превышает восьми. Например, отечественной промышленностью выпускаются восьмиразрядные микро-ЭВМ серии К1816 и четырехразрядные микро-ЭВМ серий КР1814, КР1820 и др.

Рассмотрим на примере ОМЭВМ серии КР1820 возможности и особенности применения однокристальных

микро-ЭВМ для решения практических задач.

Однокристальные микро-ЭВМ серии К1820 выпускаются промышленностью в двух модификациях: КР1820ВЕ1 — с внешним ПЗУ программ и КР1820ВЕ2 с внутренним масочным ПЗУ программ емкостью 1К байт.

Микро-ЭВМ КР1820ВЕ1 целесообразно использовать в изделиях с малой тиражностью и для отладки программного обеспечения ОМЭВМ КР1820ВЕ2. Например для изделий с большой тиражностью лучше использовать микро-ЭВМ КР1820ВЕ2. В таком случае будут оправданы экономические затраты, связанные с проектированием и изготовлением фотошаблонов и ПЗУ, соответствующих рабочей программе.

Для практического применения ОМЭВМ в решении прикладной задачи не обязательно быть специалистом по вычислительной технике. Достаточно иметь общие представления о возможностях ОМЭВМ, ее структуре, ознакомиться с системой команд и научиться решаемую задачу представить в виде программ, пользуясь командами ОМЭВМ.

Для использования ОМЭВМ нужно разработать принципиальную электрическую схему проектируемого устройства, в которой по определенным правилам соединить ОМЭВМ с источниками входных сигналов и объектами управления. Разработчику такой схемы необходимо ознакомиться с внутренней структурой ОМЭВМ, ее входными и выходными сигналами и временны́ми диаграммами работы. Все эти данные приведены в разделе «Структура ОМЭВМ».

Основной объем работ при практическом использовании ОМЭВМ приходится на разработку и отладку программного обеспечения, дающего возможность работы ОМЭВМ в соответствии с исходным заданием. Для выполнения этого этапа работы необходимо изучить систему команд ОМЭВМ и ее кодировку на языке АССЕМБЛЕРа, приведенные в разделе «Система команд ОМЭВМ».

Отладка написанного программного обеспечения проводится, как правило, с помощью специальных программ, написанных для других, широко распространенных ЭВМ (назовем их отладочны́ми). С помощью этих программ можно проанализировать поведение ОМЭВМ при исполнении составленной для нее программы. Анализ поведения ОМЭВМ позволяет выполнять ее программно-логическая модель, написанная в отладочной ЭВМ и отражающая все блоки и внутренние связи ОМЭВМ. Подавая на входы такой модели коды, соответствующие выполняемым командам и входным сигналам, с помощью отладочной ЭВМ можно проанализировать изменение состояния любого блока в процессе исполнения написанной программы.

После того как работа с отладочной ЭВМ закончена и программа отлажена, ее можно записать в программную память ОМЭВМ и приступить к анализу ее работы в реальном времени с подключенными реальными устройствами. Между работой отлаживаемой программы в отладочной ЭВМ и в реальной модели могут появиться различия. Они объясняются тем, что в отладочной ЭВМ не всегда возможно учесть особенности всех подключаемых к ОМЭВМ устройств, задержки входных и выходных сигналов при обмене информацией, отклонения реальных характеристик внешних устройств от заложенных в программе. Поэтому на этапе отладки программы в реальном времени целесообразно воспользоваться эмулятором ПЗУ, в который записывается рабочая программа. С помощью той же отладочной ЭВМ рабочая программа, отлаженная на программно-логической модели, записывается в эмулятор ПЗУ, который имитирует работу программного ПЗУ ОМЭВМ.

В том случае, когда работа ОМЭВМ не соответствует заданному алгоритму, проводится корректировка рабочей программы и повторная ее отладка в реальном времени на отладочной ЭВМ и с ОМЭВМ. Программу можно

считать отлаженной, если ОМЭВМ работает в соответствии с исходным заданием. Тогда рабочую программу уже можно записать в ПЗУ, которое является неотъемлемой частью ОМЭВМ в конкретном устройстве.

Структура ОМЭВМ

Структурная схема однокристальной микро-ЭВМ КР1820ВЕ1 приведена на рис. 1. Операции над данными выполняются в четырехразрядном арифметико-логическом устройстве (АЛУ). Результат выполненной операции записывается в регистр А (аккумулятор), а возможная единица переноса записывается в триггер переноса С. Содержимое С-триггера может быть переписано в SKL-триггер, который формирует выходной сигнал SK. В качестве входных операндов АЛУ может использовать данные, хранящиеся в аккумуляторе, в ОЗУ и коды операндов исполняемых команд. АЛУ выполняет операции сложения с учетом и без учета переноса, операции пересылки, инкрементирования и декрементирования содержимого Bd-регистра в ходе выполнения операций, связанных с обращением к ОЗУ, и операцию «логическое ИЛИ». ОЗУ организовано в виде четырех регистров, каждый из которых содержит 16 четырехразрядных ячеек (рис. 2). Адрес выбранного столбца определяется содержимым регистра Bd, а адрес строки — регистром Br. Содержимое выбранной ячейки может пересылаться в аккумулятор, Q-регистр, в L-порт. К ячейкам ОЗУ может быть выполнено прямое обращение (команды IDD, XAD), при котором в коде команды содержится

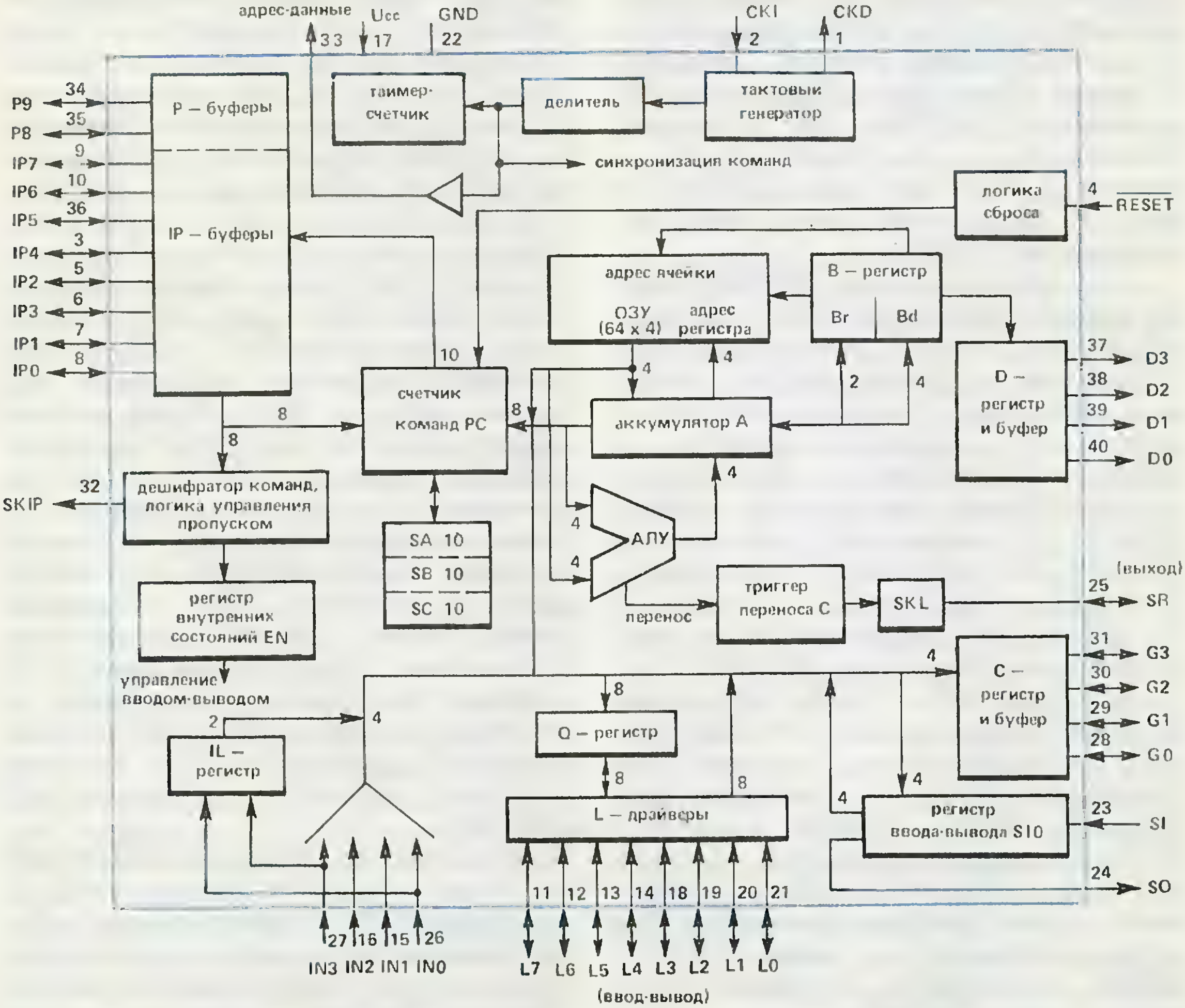


Рис. 1

Рис. 2

непосредственный адрес ячейки.

Регистр Bd можно использовать для непосредственной передачи содержимого ячейки ОЗУ в порт D.

Для обмена с внешними устройствами используется Q-регистр, имеющий двунаправленный обмен с L-портом ввода-вывода. При выполнении соответствующих команд L-драйверы выводят содержимое Q-регистра на L-порт. По другим командам информация L-порта может быть переписана непосредственно в аккумулятор и ОЗУ. L-порт имеет высокую нагрузочную способность, его можно использовать для непосредственного управления сегментами мультиплексируемого семисегментного дисплея.

Двунаправленный G-порт обслуживается четырехразрядным G-регистром и обеспечивает ввод-вывод информации.

Со входов IN данные могут заноситься в аккумулятор (команда ININ). С помощью регистра защелки IL данные с внешних устройств могут приниматься в асинхронном режиме.

Режим работы многих внутренних блоков задается кодом четырехразрядного регистра внутренних состояний EN, загружаемого командой LEIY (Y — код загружаемого слова). Младший разряд этого регистра (EN_0) определяет режим работы последовательного порта ввода-вывода SIO. При $EN_0=1$ регистр SIO работает в режиме двоичного счетчика, содержимое которого декрементируется с каждым переходом сигнала на SIO-входе с высокого уровня на низкий. Каждый входной импульс, поступающий на SI вход, должен быть по длительности не менее двух командных циклов. Состояние SO выхода в этом режиме определяется состоянием старшего разряда EN_3 регистра EN.

При $EN_0=0$ регистр SIO работает как четырехразрядный регистр, сдвигающий информацию влево с каждым командным циклом. Информация, поступающая на вход SI в этом режиме, записывается в младший разряд SIO. Состояние $EN_0=0$ при условии SKL=1 организует выдачу через SK логически управляемых синхроимпульсов. Старший разряд EN(EN_3) совместно с EN_0 управляет работой SO выхода в соответствии с табл. 1.

Второй разряд регистра EN_0 (EN_1) разрешает прерывания на входе IN_1 при $EN_1=1$. После подтверждения прерывания EN_1 принимает значение $EN_1=0$, запрещая следующие прерывания.

Разряд EN_2 управляет передачей информации через L-порт. При $EN_2=1$ информация из Q-регистра поступает в L-порт, при $E_2=0$ L-драйверы переводят L-порт в третье состояние (с высоким выходным сопротивлением).

Во времени микро-ЭВМ синхронизируется импульсами тактового генератора, которая может работать в режимах внутренней и внешней генерации. В первом случае ко входам СКО, СКІ подключается кварцевый резонатор, во втором — тактовые импульсы подаются на вход СКІ. Частота тактового генератора обычно в диапазоне 1,6—4 МГц. Время исполнения команды (командный цикл) определяется 16 периодами тактового генератора.

Таблица 1

Зависимость режимов функционирования регистра SIO и последовательного порта ввода-вывода от состояния EN-регистра

EN3	EN0	Режим работы SIO-регистра	Состояние SI-входа	Состояние SO-выхода	Состояние SKL- и SK-выходов
0	0	Сдвиговый регистр	Вход сдвигового регистра	0	При SKL = 1 SK-системная синхронизация, При SKL=0 SK=0
1	0	То же	То же	Последовательный вывод	То же
0	1	Двоичный счетчик	Вход двоичного счетчика	0	При SKL=1 SK=0 При SKL = 0 SK=0
1	1	То же	То же	1	То же

Адрес очередной выполняемой команды рабочей программы определяется кодом программного счетчика (PC) — счетчика команд. Начальная установка PC на нулевой адрес производится по входу RESET логикой сброса.

При выполнении линейных участков программ содержимое PC увеличивается с каждым командным циклом на единицу. Специальными командами переходов в PC могут быть записаны адреса команд, к которым выполняется программный переход (JMP, JSP).

Рабочая программа ОМЭВМ хранится в отдельном ПЗУ, которое информационными выводами соединяется с IP-портом. Адресный код очередной команды, сформированный в PC, передается через тот же порт IP, который используется для ввода в ОМЭВМ кода считанной команды. Для обеспечения передачи, адреса и кода команды через один и тот же порт используется схема с регистром (рис. 3), в который записывается по управляющему сигналу адрес команды.

Управляющий сигнал AD/DATE обеспечивает по перепаду с высокого уровня в низкий запись адреса команды, сформированного ОМЭВМ во время единичного уровня сигнала AD/DATE. При нулевом уровне сигнала AD/DATE порт IP переводится в режим приема кода команды, передаваемой из ПЗУ, на адресный вход которого подан код PC. Временные диаграммы, отражающие последовательность работы ОМЭВМ, приведены на рис. 4.

Считанный код команды поступает на дешифратор команды, который формирует управляющие импульсы для всех узлов ЭВМ, участвующих в выполняемой команде. Одновременно

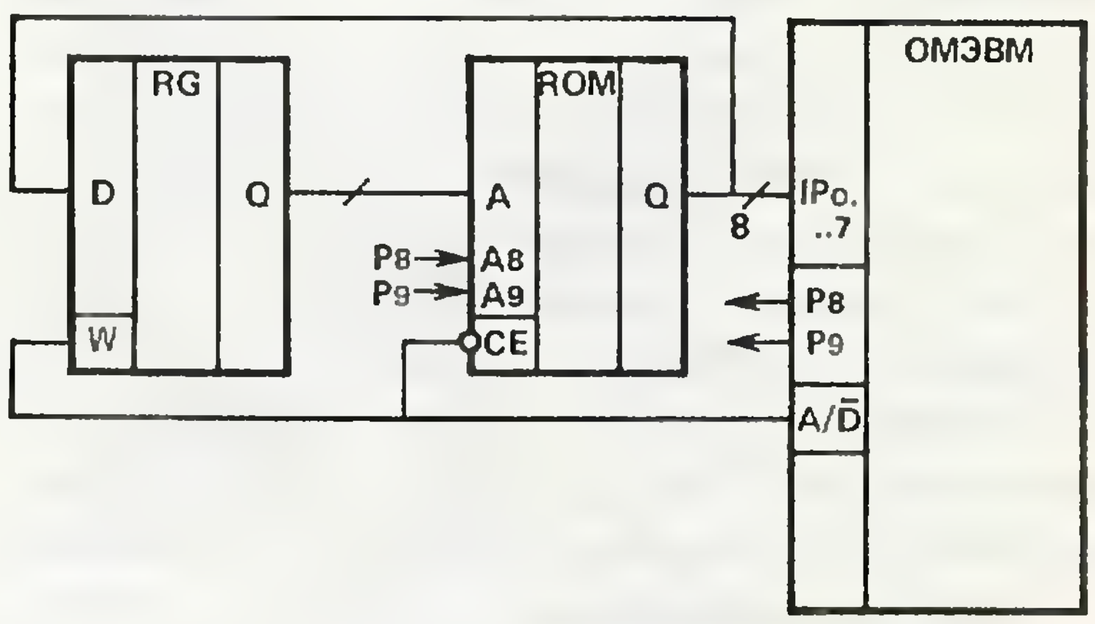


Рис. 3

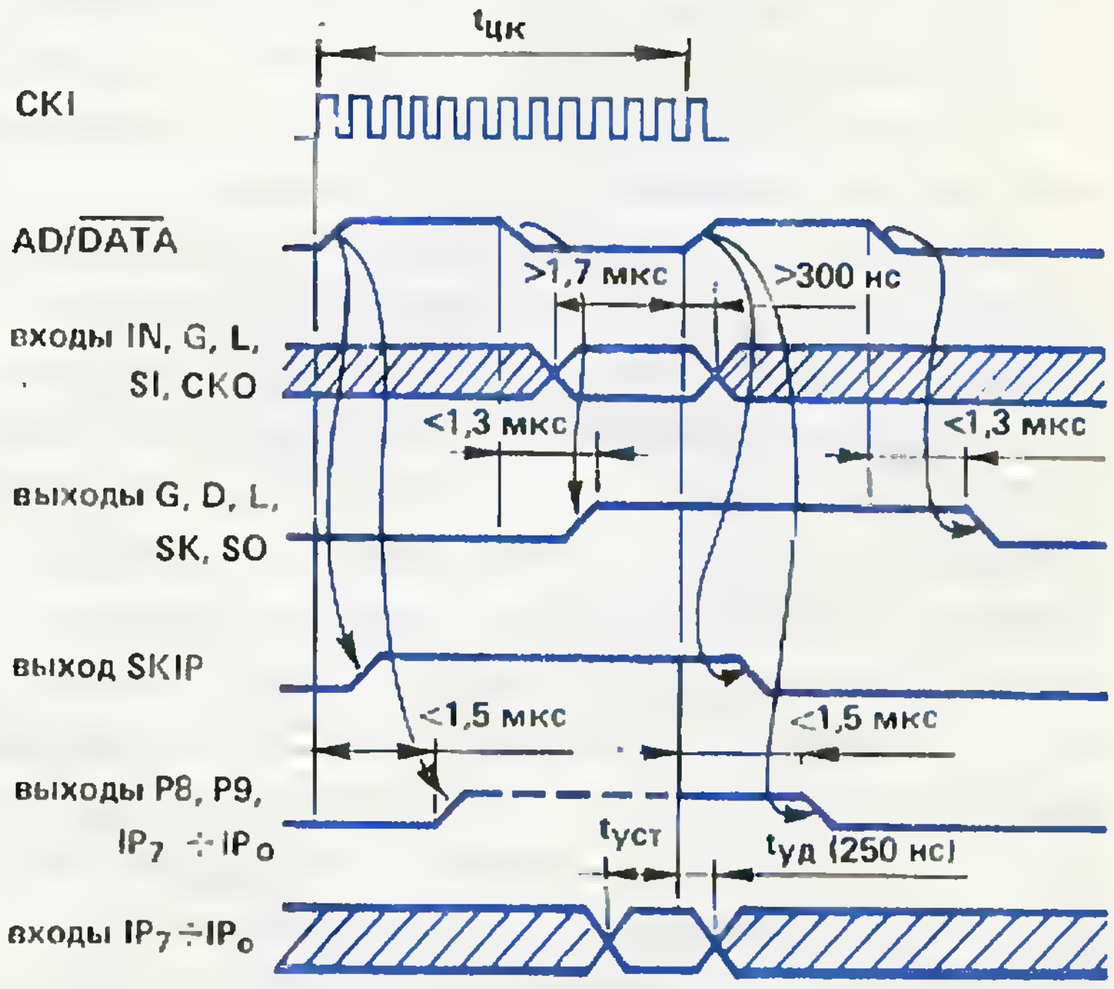


Рис. 4

с исполнением текущей команды увеличивается на единицу код PC, и процесс повторяется.

Все адресное пространство ПЗУ условно может быть разбито на 16 страниц по 64 слова. Каждые четыре страницы образуют блок объемом 256 слов. Такая организация памяти обусловлена особенностями выполнения команд условного перехода. Например, команда JP обеспечивает переход внутри страницы. Команда JSRP обеспечивает переход из любого адреса к адресу, расположенному на странице 2, что позволяет использовать эту страницу для написания подпрограмм.

В ОМЭВМ предусмотрен режим прерывания программы. В качестве запроса прерываний используется вывод IN_1. Прерывание возможно после выполнения текущей команды в случае, если в регистре EN в первом разряде установлен код 1 ($EN_1=1$). Сигнал запроса прерывания должен удерживаться на входе IN_1 не менее двух командных циклов.

Для возврата к основной программе после выполнения подпрограммы используется трехуровневый десятиразрядный стек (SA, SB, SC), который обеспечивает три уровня вложения подпрограмм. С переходом к прерывающей программе в стек записывается увеличенное на единицу содержимое счетчика команд, а записанная ранее информация опускается на нижние уровни ($PC+1 \rightarrow SA \rightarrow SB \rightarrow SC$). Записанный ранее в SC код теряется. После записи в стек адреса ($PC+1$) в программный счетчик заносится адрес последнего слова третьей страницы ПЗУ (OFF), по которому должна быть записана первая команда подпрограммы обработки прерывания. Запрет следующего прерывания достигается тем, что первый разряд регистра EN (EN_1) сбрасывается в ноль.

Для организации повторного прерывания непосредственно перед командой выхода из подпрограммы необходимо ввести команду, устанавливавшую код 1 в первом разряде регистра EN (LETI).

В ОМЭВМ имеется встроенный таймер — счетчик, который представляет собой десятиразрядный счетчик импульсов тактового генератора. При переполнении счетчика после прихода 1024×16 тактовых импульсов специальный D-триггер переключается в единичное состояние. Командой анализа кода таймера (SKT) проверяется состояние этого D-триггера, и в случае если он находится в единичном состоянии, программным счетчиком пропускается следующая команда и D-триггер переключается в нулевое состояние. Подсчитывая число переключений и зная частоту тактового генератора, можно задавать программно определенные временны́е интервалы.

Система команд ОМЭВМ

Практическое использование ОМЭВМ требует написания в ПЗУ цифровых кодов, соответствующих рабочей программе. Однако работать с программой, оформленной в виде последовательности цифр, трудно, так как, представленная таким образом, она плохо воспринимается человеком и не позволяет без специальной подготовки провести ее анализ, выполнить корректировку. Значительно удобней работать с текстом программы, составленным из команд, обозначенных буквенно-цифровыми символами и раскрывающих содержание выполняемых операций. Перевод программ с принятых буквенно-цифровых кодов в цифровые производится специальной программой ассемблер, которая может быть написана для любой управляющей или универсальной ЭВМ.

Все команды разбиты на четыре группы: арифметические, команды передачи управления, обращения к ОЗУ и проверки состояний.

Мнемоника команд представлена аббревиатурой от принятых полных названий команд на английском языке так, как это делается в большинстве алгоритмических языков программирования. Рассмотрим эти команды.

Арифметические команды.

ASC (ADD AC TO STORE AND CARRY) — сложить AC с содержи-

мым ячейки ОЗУ, адрес которой определяется регистрами Br и Bd. При выполнении этой команды формируется сигнал переноса, который может использоваться для организации условного перехода в программе. Если C=0, выполняется следующая команда программы, если C=1, организуется пропуск следующей команды и управление передается второй команде после исполняемой.

ADD суммирует содержимое AC с содержимым ОЗУ.

ADT (ADD TO TEN) — сложить с десятью.

AISC (ADD IMMEDIATE, SKIP IF CARRY) — сложить непосредственно операнд кода команд с AC, пропуск при C=1).

CASC (COMPLEMENT AC, SKIP IF CARRY) — вычитание AC из суммы RAM (B)+C, пропуск при C=1.

CLRA (CLEAR ACC) — очистить AC.

COMP (COMPLEMENT) — дополнение, инверсия.

NOP (NOT OPERATION) — пустая операция.

RC (RE SET CARRY) — сброс переноса.

SC (SET CARRY) — установка переноса.

XOR (EXCLUSIVE OR) — исключающее ИЛИ.

Команды передачи управления.

JID (JUMP IMMEDIATE DATA) — переход по непосредственно определенному адресу. Разряды 7:0 программного счетчика определяются ячейкой ПЗУ с адресом (PC 9,8, AC, RAM(B).

JMP.. (JUMP) — переход по адресу, в котором разряды PC9, PC8 определяются кодом y2, y3, а PC 7 ... 0 кодом XX.

JP (JUMP IN PAGE) — переход в пределах страницы.

JSRP (JUMP TO SUBROUTINE IN PAGE) — переход к подпрограмме в пределах страницы.

JSR (JUMP TO SUBROUTINE) — переход к подпрограмме.

RET(RETURN) — возврат из подпрограммы.

RETSK (RETURN WITH SKIP) — возврат с пропуском.

Команды обращения к ОЗУ

CAMO(CONVEY AC AND MEMORY TO Q) — передать содержимое AC и ОЗУ в Q-регистр.

CQAM — обратная предыдущей операции.

LD (LOAD DATA) — загрузить данные в AC.

LDD — аналогична предыдущей операции.

LQID (LOAD Q IMMEDIATE DATA) — загрузить Q-регистр непосредственными данными из ПЗУ.

RMB (RESET MEMORY BIT) — установить в «0» бит ОЗУ.

SMB (SET MEMORY BIT) — установить в «1» бит ОЗУ.

STIT (SET IMMEDIATE INFORMATION) — установить непосредственную информацию.

X (EXCHANGE) — обмен.

XAD (EXCHANGE AS WITH DATA)— обмен информацией AC с ОЗУ.

XDS (EXCHANGE AND DECREMENT STORE) — обмен с уменьшением адреса памяти.

XIS (EXCHANGE INCREMENT STORE) — обмен с увеличением адреса памяти.

CAB (CONVEY AC TO B) — переслать содержимое AC в B.

CBA — обратная предыдущей команде.

LBI (LOAD B IMMEDIATE) — загрузить B непосредственно.

LEI (LOAD EN IMMEDIATE) — загрузить EN непосредственно.

XABr (EXCHANGE AC TO Br) — обмен AC с Br.

Команды проверки состояний

SKC (SKIP IF C) — пропуск при C=1.

SKE (SKIP IF EQUAL) — пропуск при равенстве.

SKGZ (SKIP IF G=ZERO) — пропуск при G=0.

SKVBZ (SKIP IF MEMORY BIT=0)— пропуск при равенстве 0 бита памяти.

SKT (SKIP IF TIMER) — пропуск при переполнении таймера.

ING (INPUT G) — ввод с порта G.

ININ (INPUT IN) — ввод с порта IN.

INIL (INPUT IL) — ввод с порта IL данных 3 и 0 разрядов.

INL (INPUT L) — ввод с порта L.

OBD (OUTPUT B TO D) — вывод B в D-порт.

OGI (OUTPUT TO G IMMEDIATE) — вывод в G-порт непосредственно из команды.

OMG (OUTPUT MEMORY TO G) — вывод данных памяти в G-порт.

XAS (EXCHANGE AS TO SIO) — обмен AC с портом SIO.

Составление программ для ОМЭВМ

Допустим, что нам необходимо составить подпрограмму для управления дисплеем, состоящим из четырех семисегментных индикаторов. Обычно для этих целей используется динамический режим индикации, суть которого сводится к следующему. Все соответствующие сегменты индикаторов объединены едиными магистралями, по которым передаются сигналы порта L, обеспечивающие высвечивание каждым индикатором своего знака (рис. 5). В каждый момент времени включенным может быть только один индикатор. Именно для этого индикатора через порты в этот момент передается управляющая информация. При выводе информации индикаторы включаются последовательно с такой частотой, что создается впечатление, что включены все индикаторы одновременно.

В нашем примере номер включаемого индикатора ОМЭВМ определяется выходным кодом порта ОМЭВМ D, сигналы, управляющие включением сегментов, передаются через порт L.

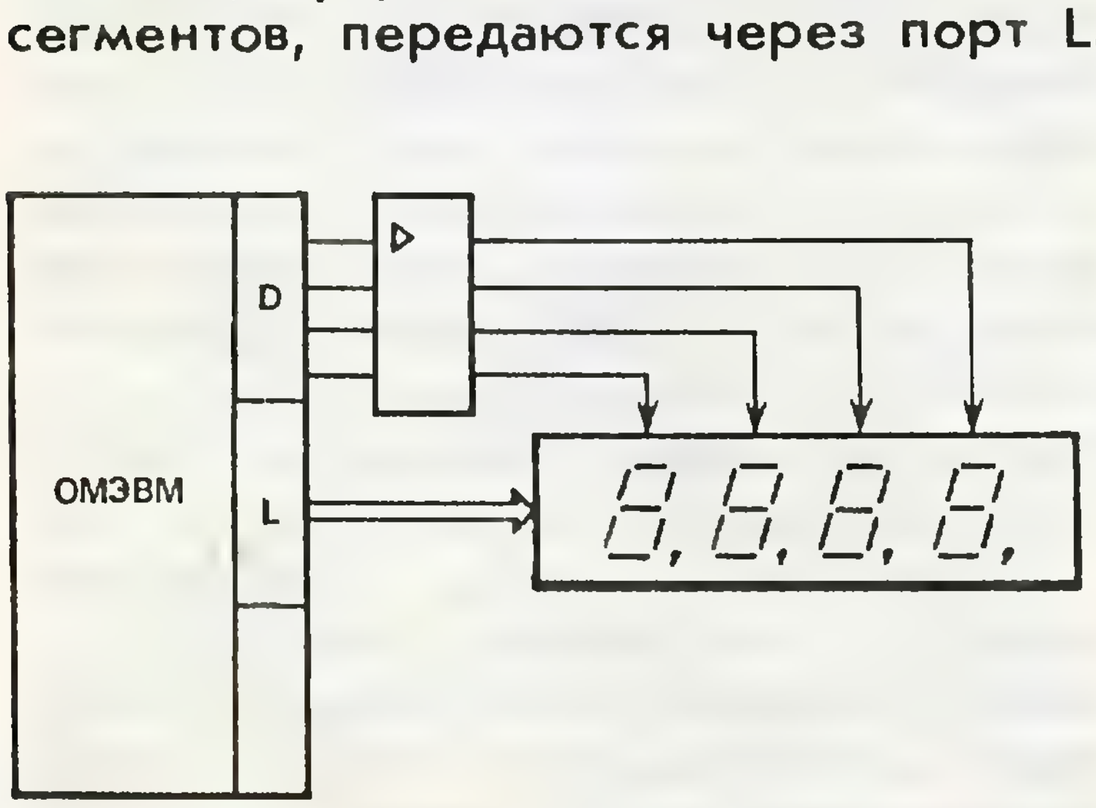


Рис. 5

Со сменой кода на выходе порта D ОМЭВМ изменяет код выходной информации порта L, обеспечивая вывод на каждый индикатор соответствующего ему символа.

Двоичные коды выводимой информации хранятся в определенных ячейках ОЗУ. Допустим, что коды выводимых символов записаны в ячейках ОЗУ с адресами Br. Bd: 3.1, 3.2, 3.3, 3.4.

Причем число, записанное в ячейке ОЗУ с адресом 3.1, должно выводиться на первый разряд индикатора, число, записанное в ячейке ОЗУ с адресом 3.2, — на второй разряд и т. д. Для индикации содержимого ячеек ОЗУ необходимо преобразовать коды записанных чисел в управляющие сигналы, включающие сегменты, соответствующие выводимому числу.

Преобразование выводимых двоичных кодов в управляющие сигналы для семисегментных индикаторов может выполняться с использованием таблицы кодировки, записанной в ПЗУ. Предположим, что, начиная с адреса 040_{16}, по адрес $04F_{16}$ в ПЗУ записаны коды, обеспечивающие индикацию символа, совпадающего с последней цифрой адреса в 16-теричной системе счисления. Таким образом, по адресу 040_{16} в ПЗУ записан код, обеспечивающий вывод цифры «0», по адресу 047_{16} — код, обеспечивающий вывод цифры «7», и т. д.

Алгоритм подпрограммы индикации представлен блок-схемой (рис. 6). В соответствии с блок-схемой составляется рабочая программа на языке АССЕМБЛЕРа. Исходный текст программы состоит из трех видов информации: меток, команды в кодах ассемблера и комментариев. Метки используются для обозначения команд, к которым в процессе исполнения программ организуются переходы. Комментарии отделяются от кодов команд знаком «;» и обычно содержат произвольную информацию, помогающую понимать выполняемые командой действия. Исходный текст программы (табл. 2) вводится в ЭВМ и обрабатывается программой Ассемблер.

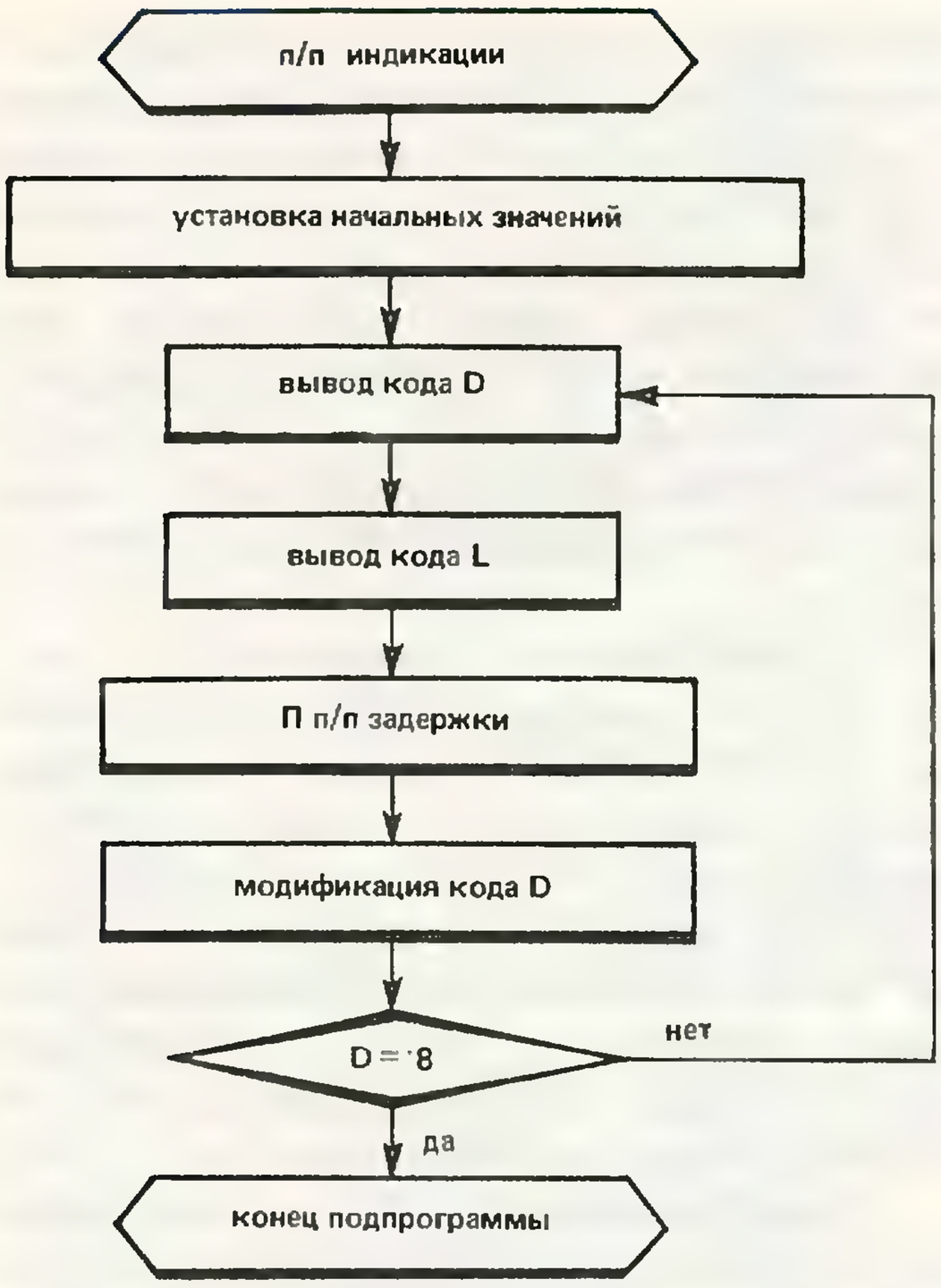


Рис. 6

Таблица 2

```
Метка    Команда      ;  Комментарий
IND:     LBI 3,0      ;  3,0=>Br, Bd
         STII 1       ;  1=>RAM(3,0), Bd+1=Bd
ID:      OBD          ;  OUT Bd D
         CLRA         ;  0=>AC
         AISC 4       ;  AC+4=AC
         LQID         ;  ROM (PC8, 9; RAM; AC)=>Q
         LEI 4
         JSR ZDR
         LDD 3,0      ;  RAM(3,0)=>AC
         ADD          ;  AC+RAM(3,0)=>AC
         XAD 3,0      ;  AC=>Bd
         LDD 3,0      ;  RAM(3,0)=>AC
         AISC 7       ;  AC+7=AC
         JP T1
         RET
T1:      LDD 3,0      ;  RAM(3,0)=>AC
         CAB          ;  AC=>Bd
         JP ID
```

В результате такой обработки формируется программа, содержащая 16-теричные коды команд, и в процессе работы программы Ассемблер выявляются имеющиеся в исходном тексте синтаксические ошибки и оператору представляется возможность откорректировать текст (табл. 3).

Подпрограмма управления дисплеем должна последовательно обеспечить чтение содержимого ячеек ОЗУ (3.1, 3.2, 3.3, 3.4) в соответствии со считанным кодом, извлечь из ПЗУ управляющий код для индикации символа и передать его по магистрали L на определенный индикатор дисплея.

Командами 1, 2 реализуется установка начальных данных, заключающаяся в загрузке адреса 3,0 в регистр B, записи 1 по этому адресу и увеличении на 1 содержимого регистра Bd. Вывод кода для подготовки включения первого индикатора обеспечивается третьей командой.

Команды 4, 5, 6 дают возможность вывода управляющих кодов по магистрали L. Код высвечиваемого символа определяется содержимым ячейки ОЗУ с адресом 3.1. Команда LQID обеспечивает запись в Q-регистр кода, считанного из ПЗУ. Передача информации из Q-регистра в L-порт производится по команде LEI4. Свечение индикатора, начатое после команды LEI4, будет продолжаться в течение времени, определяемого подпрограммой задержки. Обращение к такой подпрограмме осуществляется командой JSR. Предполагается при этом, что имеется готовая подпрограмма задержки с меткой ZDR.

После выполнения подпрограммы задержки выполняются команды 9, 10, 11, происходит сдвиг единицы, записанной в регистр Bd, и подготовка к включению следующего индикатора.

Команды 12, 13 реализуют счетчик циклов в программе. После очередного сдвига единицы в регистре Bd его содержимое записывается в AC и к нему прибавляется код 7. В зависимости от значения кода переноса C будет выполняться 14 или 15 команда.

При C=0 команды 16, 17 пересылают содержимое ячейки ОЗУ (3,0) в регистр Bd, и после этого выполняется переход в начало программы к третьей команде, обеспечивая включение следующего индикатора.

Сигнал C=1 после выполнения команды 13 соответствует завершению вывода информации на четвертый индикатор, что определяет завершение подпрограммы индикации.

При C=1 после команды 13 будет

Таблица 3

CROSS ASSEMBLER PAGE: 1

№ команды	Адрес команды	Код команды	Метка	Команда		Комментарий
1	001	3F	IND: IND:	LBI 3,0	;	3,0=>Br, Bd
2	001	71		STII 1	;	1=>RAM(3,0), Bd+1=Bd
3	002	333E	ID:	OBD	;	OUT Bd D
4	004	00		CLRA	;	0=>AC
5	005	54		AISC 4	;	AC+4=AC
6	006	BF		LQID	;	ROM (PCB,9; RAM; AC)=>Q
7	007	3364		LEI 4		
8	009	6800		JSR ZDR		
9	00B	2330		LDD 3,0	;	RAM(3,0)=>AC
10	00D	31		ADD	;	AC+RAM(3,0)=>AC
11	00E	23BO		XAD 3,0	;	AC=>Bd
12	010	2330		LDD 3,0	;	RAM(3,0)=>AC
13	012	57		AISC 7	;	AC+7=AC
14	013	D5		JP T1		
15	014	48		RET		
16	015	2330	T1:	LDD 3,0	;	RAM(3,0)=>AC
17	017	50		CAB	;	AC=>Bd
18	018	C2		JP ID		

выполняться команда 15 возврата к основной программе.

В качестве другого примера рассмотрим подпрограмму, обеспечивающую сложение чисел, представленных в двоично-десятичном коде. Каждая ячейка ОЗУ используется для хранения одного десятичного числа. Допустим, что 12-разрядные коды слагаемых записаны в ячейках ОЗУ с адресами 00 — $0C_{16}$ и 10—$1C_{16}$, результат сложения необходимо записать в ячейки ОЗУ по адресу 00—$0C_{16}$ (адреса записаны в 16-теричной системе счисления).

Особенность выполнения операций над десятичными числами связана с необходимостью корректировки результата в случае, если он превышает 10. Эта корректировка требует вычитания из полученного результата десяти и формирования единицы переноса в следующий разряд. Вычитание десяти организуется как вычитание 16 и прибавление 6 к корректируемому результату. Вычитание 16 сводится к игнорированию переноса из четвертого разряда сумматора, который соответствует числу 16.

В соответствии с блок-схемой подпрограммы сложения (рис. 7) в первом блоке осуществляется запись в регистр Br, Bd кода 1.0 адреса младшего разряда первого слагаемого. Во втором блоке суммируются числа, записанные в ОЗУ, адреса которых определяются кодом Bd, а коды Br соответственно равны 00 и 01. Модификация кода Br выполняется командой LD 01, по которой одновременно с загрузкой Ac реализуется операция сложения по модулю 2 кода 01 и содержимого регистра Br. В случае когда результат суммирования превышает 16, управление передается четвертому блоку, в котором производится десятичная коррекция результата. Если результат суммирования меньше 16, то управление передается пятому блоку, где организуется сравнение полученного результата с числом 10. В шестом блоке реализуется коррекция суммы, если она превышала десять, а в седьмом блоке — сохранение суммы, если она не превышала десяти. Запись результата в ячейки ОЗУ, в которых было записано второе слагаемое, выполняется восьмым блоком. Одновременно с записью результата увеличивается на единицу код регистра Bd, изменяется номер разрядов суммируемых чисел. В девятом блоке Bd сравнивается с максимальным номером разряда 12, и если Bd<12, управление передается блоку 2 для суммирования следующих разрядов чисел, если же Bd=12, суммирование 12-разрядных чисел завершено.

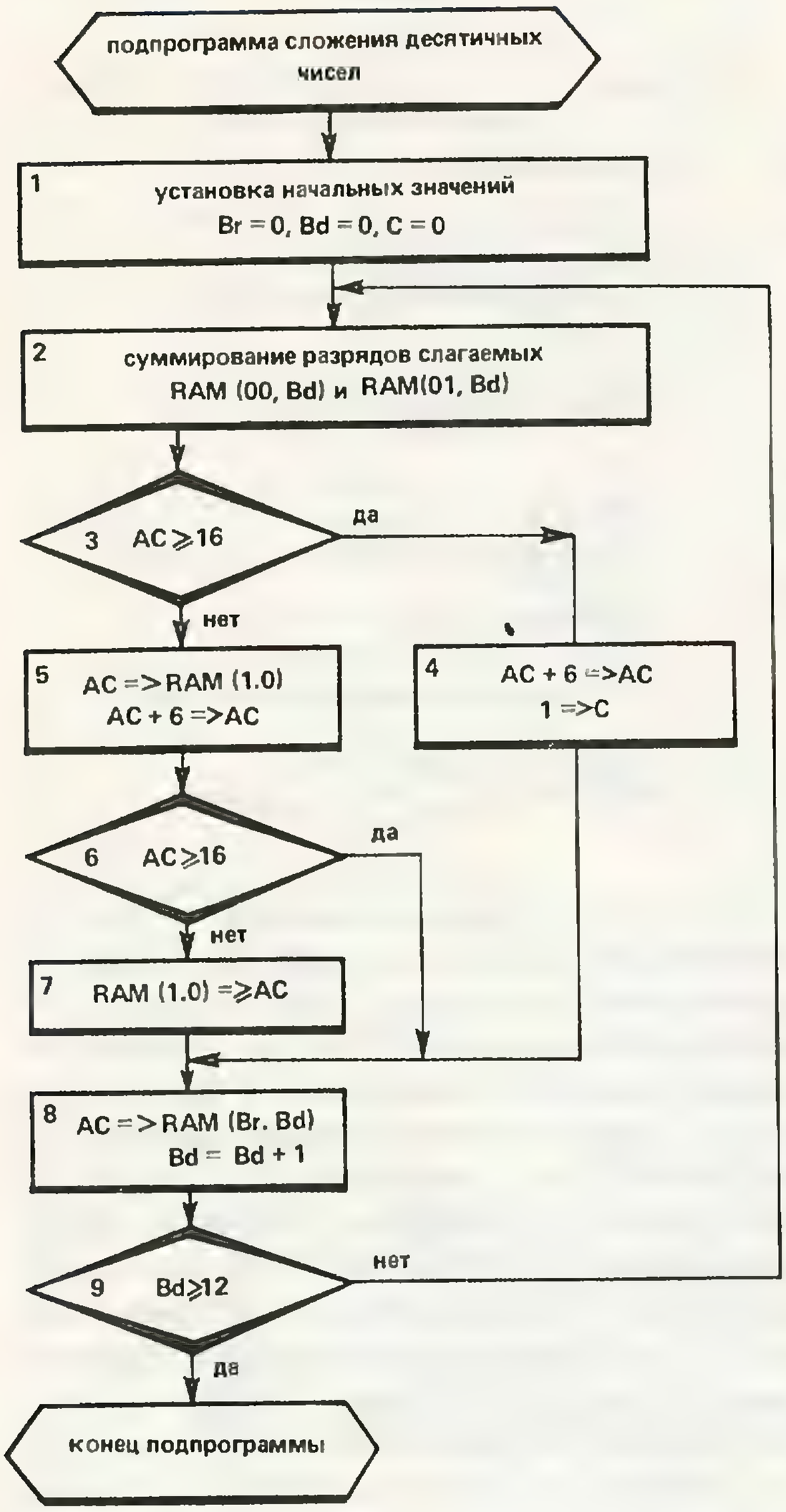


Рис. 7

Текст подпрограммы, обеспечивающей сложение десятичных чисел, полученный после обработки исходного текста программой Ассемблера, представлен в табл. 4.

Первые две команды программы сложения десятичных чисел обеспечивают загрузку начального адреса одного из слагаемых в регистр B и запись нуля в триггер C. Командой LD 01 код слагаемого переписывается в аккумулятор, а код регистра B модифицируется, указывая на адрес первого разряда второго слагаемого.

Четвертая команда обеспечивает сложение операндов и запись переноса в триггер C. При выполнении этой команды выполняется условный переход в программе: при C=1 управление передается шестой команде, а при C=0 — пятой.

В случае возникновения переноса требуется провести корректировку результата. Шестой командой к результату сложения прибавляется шесть, устанавливается единица в триггер C, при этом результат сложения находится в AC. Командой JMPT управление передается в конец программы для организации цикла, обеспечивающего последовательное сложение всех разрядов операндов.

В случае отсутствия переноса управление передается команде с меткой N. Отсутствие переноса при сложении десятичных чисел может быть вызвано двумя различными ситуациями: результат меньше десяти, и его не надо корректировать, и результат больше 10, но меньше 16, и его требуется корректировать.

Командами XAD, LDD результат записывается в ячейку ОЗУ (1,0) и переписывается в AC. Командой AISC6 к содержимому AC прибавляется шесть, и в зависимости от значения переноса C управления передается команде с меткой L или T. При отсутствии переноса (C=0) командами LDD 1,0 и XIS 1 восстанавливается в AC результат сложения, записанный в команде 10 в ОЗУ по адресу (1.0).

Команды, начинающиеся с метки T, обеспечивают запись результатов вычислений из AC в ячейку ОЗУ, где ранее был записан код первого слагаемого, и увеличение на единицу кода регистра Bd, при этом будет переход к обработке следующего разряда слагаемых.

Команды 18 и 19 определяют условия выхода из цикла. При содержимом Bd, меньшем двенадцати, управление будет передаваться в начало программы к метке ADD 10, что обеспечит последовательное суммирование двенадцати разрядов слагаемых. После подсчета результата в двенадцатом разряде управление будет передано команде RET, по которой произойдет переход к основной программе по содержимому стека.

Таблица 4

CROSS ASSEMBLER PAGE: 1

№ команды	Адрес команды	Код команды	Метка	Команда		Комментарий
1	000	1Ф		ЛБИ 1,0	;	1,0=>Br, Bd
2	001	32		RC	;	0=>C
3	002	15	ADD10:	LD 1	;	RAM(B)=>AC; Br+1=>Br
4	003	30		ASC	;	AC+C+RAM(B)=>AC, CARRY=>C
5	004	60B		JMP N		
6	006	56		AISC 6	;	AC+6=>AC
7	007	44		NOP		
8	008	22		SC	;	1=>C
9	009	6016		JMP T		
10	00B	2390	N:	XAD 1,0	;	RAM(1,0)<=>AC
11	00D	2310		LDD 1,0	;	RAM(1,0)=>AC
12	00F	56		AISC 6	;	AC+6=>AC
13	010	6014		JMP L		
14	012	6016		JMP T		
15	014	2310	L:	LDD 1,0	;	RAM(1,0)=>AC
16	016	14	T:	XIS 1	;	RAM(B)<=>AC, Bd+1=Bd
17	017	4E		CBA	;	Bd=>AC
18	018	53		AISC 3	;	AC+3=>AC
19	019	C2		JP ADD10		
20	01A	48		RET		

Отладочные комплексы

Между работой отлаживаемой программы в отладочной ЭВМ и в реальной ОМЭВМ могут появиться различия. Они объясняются тем, что в отладочной ЭВМ не могут быть учтены особенности всех подключаемых к ОМЭВМ устройств, задержки входных и выходных сигналов при обмене информацией, отклонение реальных характеристик внешних устройств от заложенных в программе. Поэтому на этапе отладки программы в реальном времени целесообразно воспользоваться эмулятором ПЗУ, в который записывается рабочая программа. Рабочая программа, отлаженная на программно-логической модели, записывается с помощью той же отладочной ЭВМ в эмулятор ПЗУ, который имитирует работу . программного ПЗУ ОМЭВМ в реальном времени. В случае если работа ОМЭВМ не соответствует заданному алгоритму, проводится корректировка рабочей программы и повторное ее опробование на отладочной ЭВМ с ОМЭВМ в реальном времени. Программу можно считать разработанной, если ОМЭВМ работает в соответствии с исходным заданием. В этом случае отлаженную рабочую программу записывают в ПЗУ, которое является неотъемлемой частью ОМЭВМ в конкретном устройстве.

Примеры применения однокристальных микро-ЭВМ

ОМЭВМ содержат набор функциональных блоков, позволяющий отнести их к практически законченным устройствам управления, которые при предельно малых габаритных размерах обладают большими вычислительными и логическими возможностями и высокой надежностью.

Выпускаемые массовым тиражом дешевые однокристальные микро-ЭВМ предназначены для использования в управлении приборами и аппаратурой культурно-бытового назначения, в контроллерах технологических процессов, измерительной аппаратуре, автомобильных контроллерах, т. е. в разнообразных системах управления и обработки данных.

При работе большинства контроллеров необходима операция аналого-цифрового преобразования. Однокристальные микро-ЭВМ дают возможность применять различные методы

преобразования, в том числе и без использования АЦП.

Рассмотрим один из таких методов — метод .широтно-импульсной модуляции, или метод рабочего цикла — на примере использования однокристальной микро-ЭВМ серии К1820 [1].

Он основан на том, что если на RC цепочку подавать повторные импульсы напряжения, то конденсатор заряжается до среднего напряжения, так как постоянная времени RC цепи достаточно велика по сравнению с периодом импульса.

Схема аналого-цифрового преобразования методом широтно-импульсной модуляции изображена на рис. 8.

В этом методе напряжение на конденсаторе U_c сравнивается с измеряемым напряжением $U_{вх}$ с помощью аналогового компаратора, а затем корректируется, чтобы приблизить его к значению $U_{вх}$. ОМЭВМ программно анализирует значение на выходе компаратора. Если U_c ниже входного напряжения (на выходе компаратора — единица), то на конденсатор с выхода G2 микро-ЭВМ для зарядки подается положительное напряжение. Если U_c выше, чем входное напряжение, то на конденсатор подается низкое напряжение для разрядки. Следовательно, значение напряжения конденсатора U_c колеблется около значения входного напряжения. Расчет показывает, что входное напряжение можно определить точно, измеряя время T1 и T2 (рис. 9).

Проблемы этого метода — необходимость точного измерения величины T1 и T2 и достаточно частых проверок состояния компаратора. Поэтому измеряются периоды многих импульсов, при этом получаемая средняя величина обеспечивает достаточную точность, но в связи с многократностью измерений значительно увеличивается время преобразования.

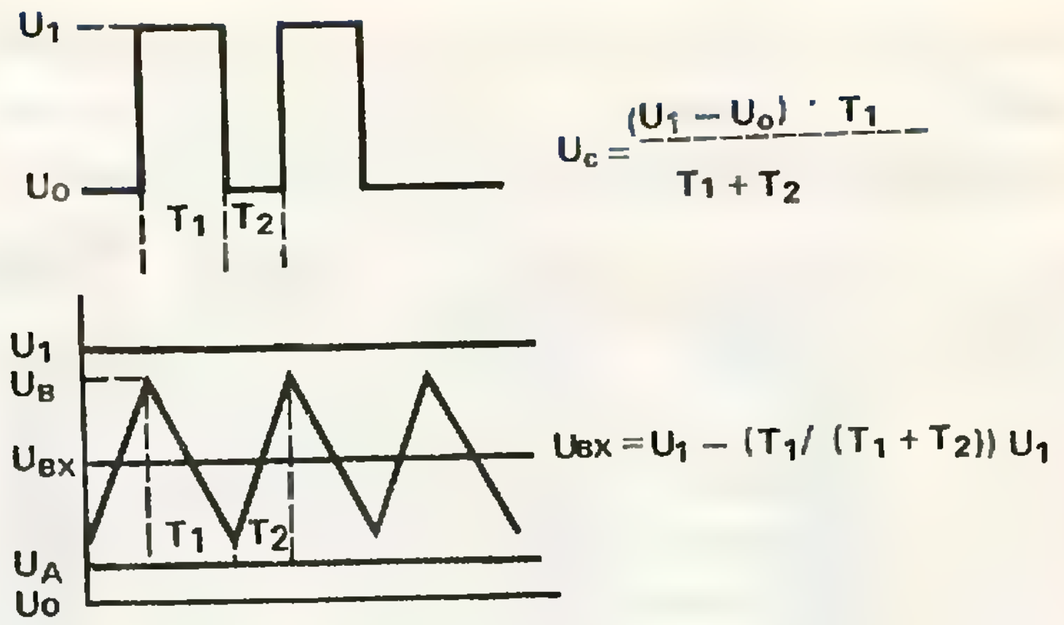


Рис. 9

Времена T1 и T2 измеряются с помощью организованных в ОЗУ двух счетчиков, один из которых считает T2, а другой — сумму T1 и T2.

Блок-схема алгоритма аналого-цифрового преобразования методом широтно-импульсной модуляции представлена на рис. 10.

Рассмотрим применение однокристальных микро-ЭВМ в контроллерах, используемых для управления различной аппаратурой бытового назначения, на примере разработок отдела электронной аппаратуры японской фирмы «Мацусита сенге кики».

Контроллеры для управления СВЧ электроплитами

Применение ОМЭВМ в контроллерах управления СВЧ электроплитами позволяет облегчить труд домохозяек путем экономии времени, затрачиваемого на приготовлении пищи, за счет частичной или даже полной автоматизации работы электроплиты. Следует также отметить, что при этом повышается качество приготовленной пищи, так как микро-ЭВМ обеспечивает строгий контроль за поддержанием заданного режима работы плиты.

Контроллеры для управления СВЧ электроплитами [Л2] обладают сле-

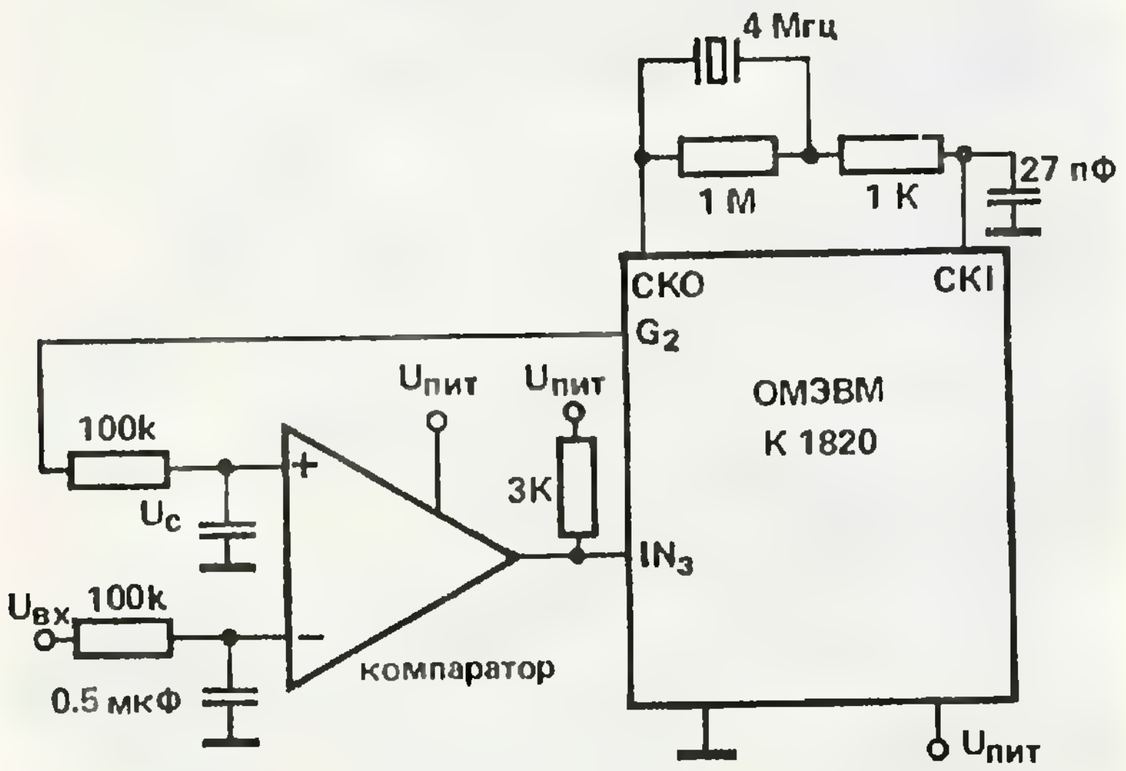


Рис. 8

дующими функциональными возможностями:

непрерывное изменение уровня выходной мощности электроплиты в пределах от 0 до 100 от предельной мощности;

программируемое приготовление еды в три этапа при помощи комбинирования уровней выходной мощности и времени нагревания на каждом этапе;

приготовление еды с использованием температурных сенсоров;

приготовление еды с использованием сенсоров влажности;

приготовление еды в режиме автостарта;

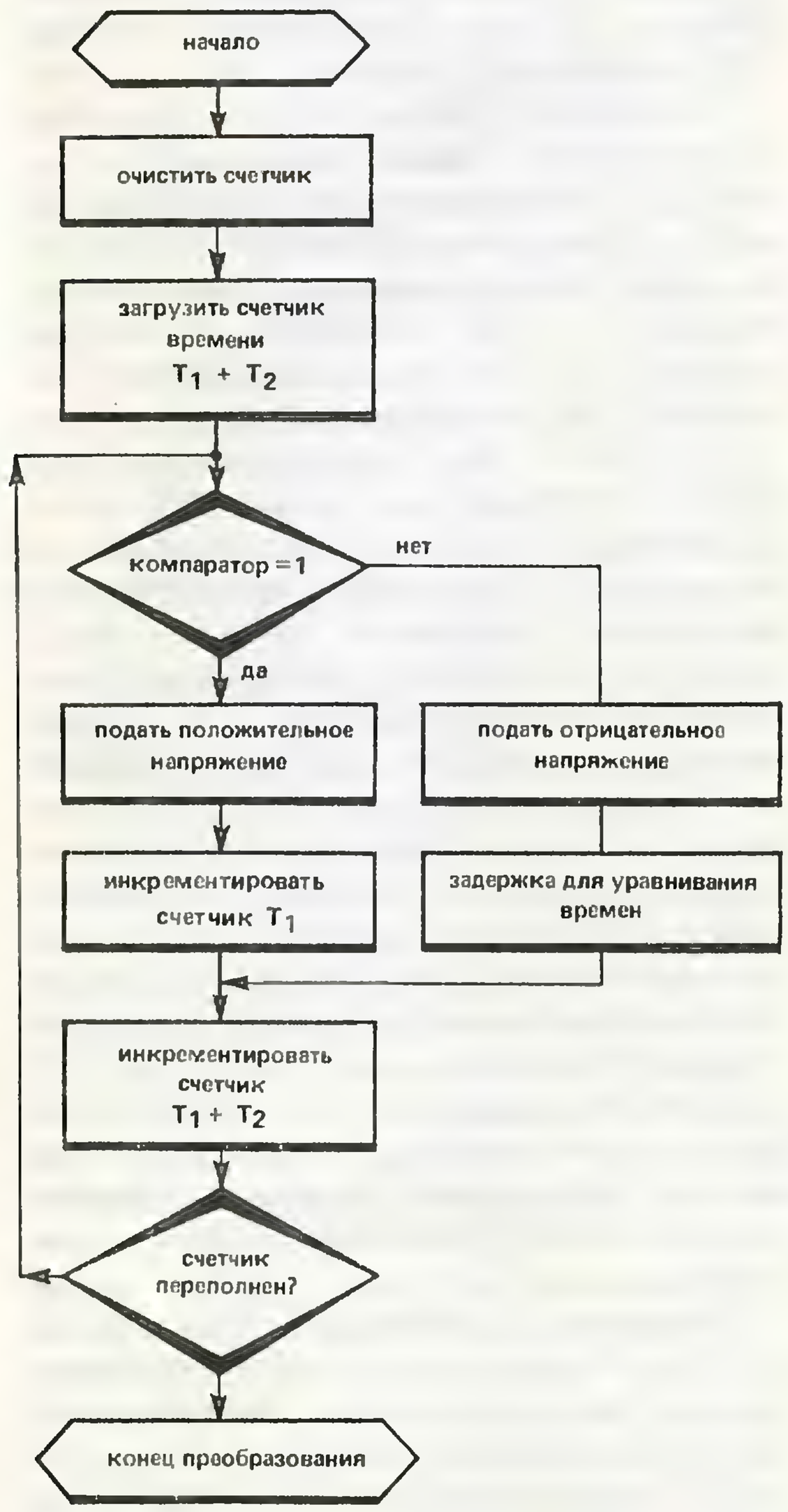


Рис. 10

приготовление еды при помощи карты меню;

возможность использования часов и таймера;

возможность многоразрядного цифрового отображения информации и индикации заданных условий работы;

возможность ввода данных при помощи клавиатуры панели ручного управления.

Рассмотрим некоторые режимы работы контроллера СВЧ электроплит: программное приготовление еды в три этапа; приготовление еды при помощи сенсоров температуры и влажности.

Временные диаграммы этих режимов представлены на рис. 11.

При программном приготовлении еды в три этапа с пульта ручного управления вводятся значения уровней выходной мощности плиты P1, P2, P3, которые необходимо получить на каждом этапе, и время нагревания до заданного уровня (T1, T2, T3). В соответствии с различными видами приготовления еды составляются различные комбинации (P1, T1), (P2, T2), (P3, T3).

При приготовлении еды с помощью сенсоров температуры (рис. 11 б) в пищу вводятся температурные пробники. Приготовление пищи можно выполнять, задавая при помощи клавиатуры необходимую температуру. Есть возможность поддерживать температуру на определенном уровне.

С применением ОМЭВМ возросли функциональные возможности контроллеров, но, с другой стороны, ухудшилась операционность системы. Авторы разработок видят выход из этой проблемы в применении системы сенсоров влажности, что приведет к автоматизации работы СВЧ электроплиты и к разработке новых улучшенных сенсоров.

Процесс приготовления еды при помощи сенсоров влажности (рис. 11в) также можно разделить на три этапа: T1 — время подготовки, необходимое для очистки сенсора; T2 — время до обнаружения пара, образую-

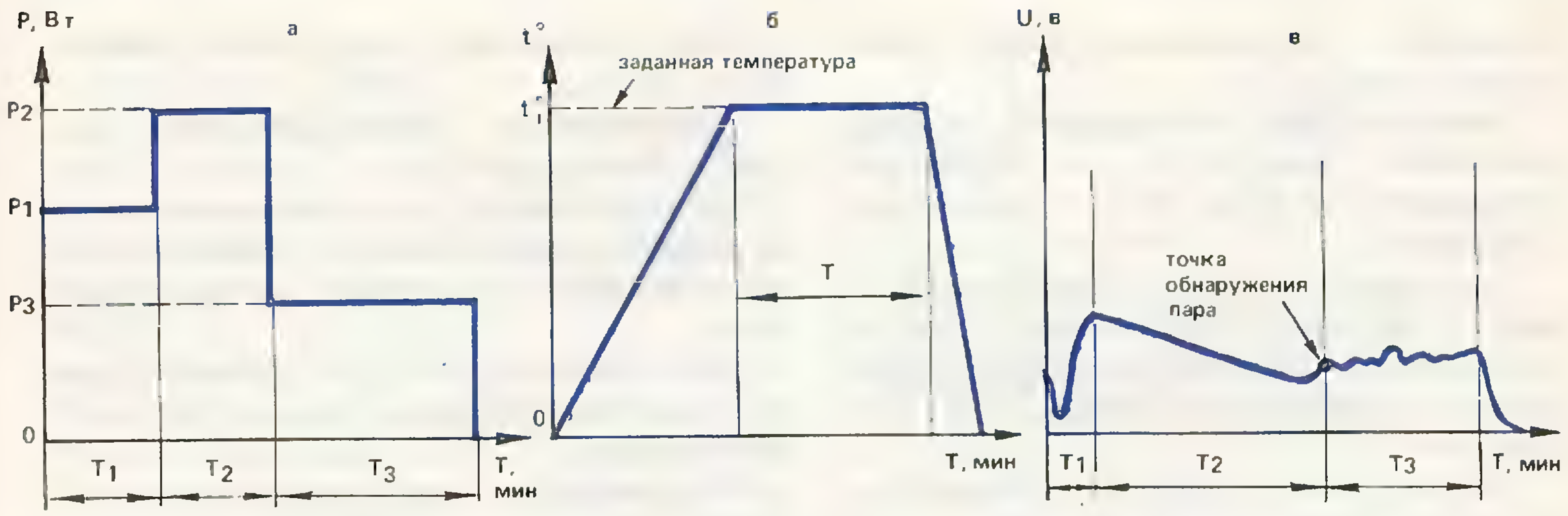


Рис. 11

щегося при нагревании пищевого продукта; Т3 — время завершения приготовления пищи, причем последнее определяется по формуле $T3 = K_ж T2$. Время Т2 определяется количеством пищевого продукта, коэффициент $K_ж$ определяется эмпирически, в зависимости от продукта; и его значение занесено в ПЗУ ОМЭВМ. Микро-ЭВМ заканчивает приготовление пищи, автоматически вычисляя время Т3.

Структура контроллера для СВЧ электроплиты показана на рис. 12. Микро-ЭВМ представляет собой однокристальный микрокомпьютер. Объем ПЗУ — 2—4 Кбайт, объем ОЗУ — 256—512 бит.

Требуемый режим работы контроллера можно задать вручную с помощью сканируемой клавиатуры панели управления. Программы работы контроллера в каждом режиме находятся в ПЗУ ОМЭВМ, исходные данные (Р, Т — и т. д.), вводимые с пульта, заносятся в ОЗУ.

Контроль за процессом работы СВЧ электроплиты осуществляется с помощью сенсоров температуры и влажности. Для этого производится сравнение выходного сигнала ЦАП, соответствующего определенному значению температуры или влажности, с выходным сигналом сенсора температуры или логарифмического усилителя сенсора влажности. В зависимости от полученных значений микро-ЭВМ вырабатывает сигналы управления СУ для обеспечения изменения уровня выходной мощности электроплиты.

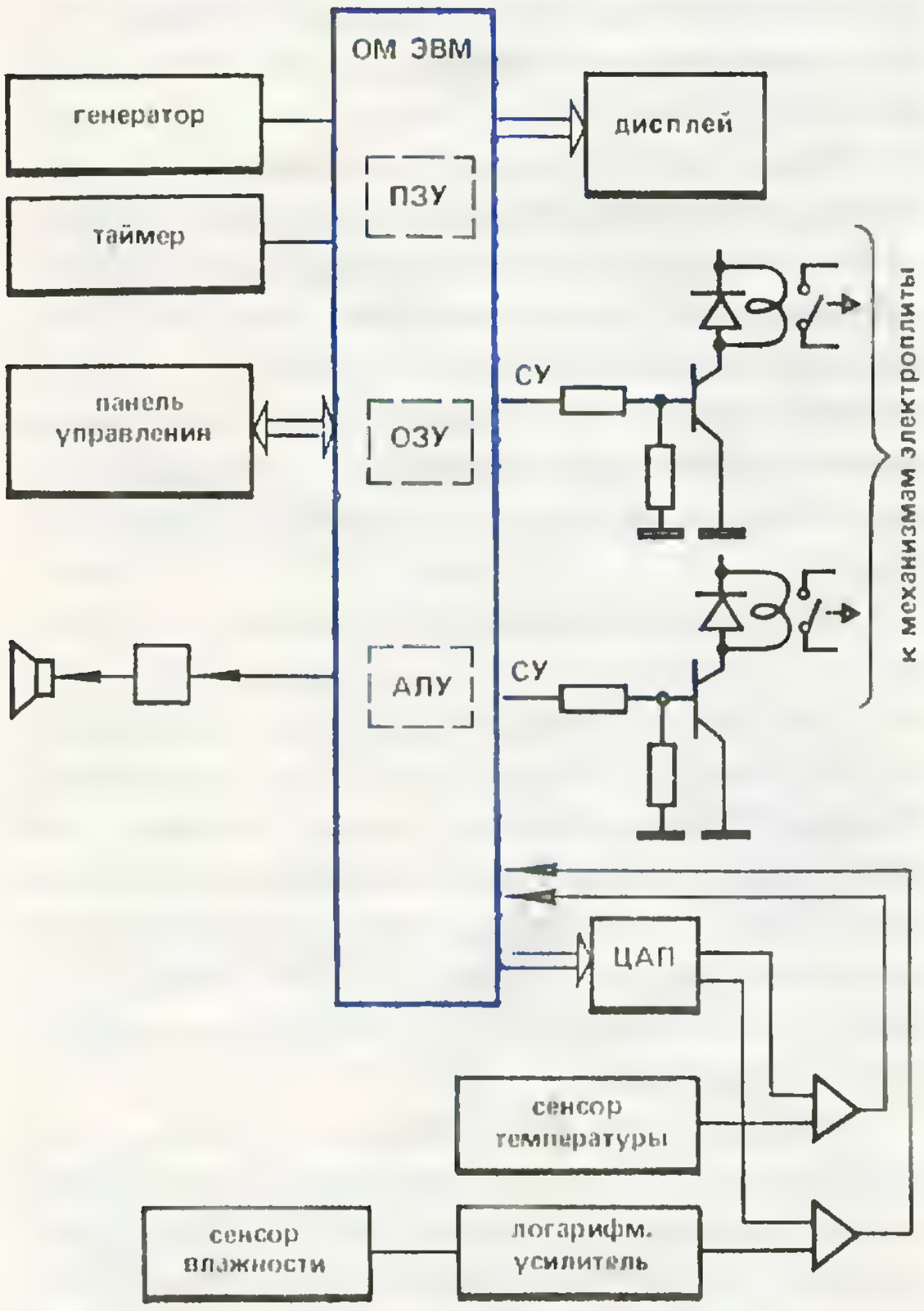


Рис. 12

В ряде режимов при формировании управляющих сигналов учитываются показания таймера (например, при приготовлении пищи в три этапа).

Условия работы электроплиты (Р, Т, а также время) отображаются на дисплее с помощью семисегментных индикаторов.

По окончании процесса приготовления пищи по команде микро-ЭВМ раздается звуковой сигнал.

Контроллеры для стиральных машин

Применение ОМЭВМ в контроллерах управления стиральными машинами [2] позволяет облегчить и ускорить выполнение трудоемких операций стирки, полоскания и отжима белья.

В полностью автоматизированной стиральной машине последовательно выполняются следующие операции:

1. Открывается клапан снабжения водой, начинается подача воды; переключатель уровня воды прекращает подачу воды.

2. В процессе стирки задается один из двух уровней потока воды — сильный или слабый. В случае сильного потока чередуются направления вращения потока — вправо и влево.

3. Откачка воды производится по истечении определенного времени после установки переключателя уровня воды в положение «выключено».

4. Обезвоживание производится в одном из двух режимов: простом или пульсирующем.

5. Процесс полоскания начинается, когда переключатель уровня воды установлен в положение «включено». После самой последней операции раздается сигнал зуммера.

6. В конце процесса стирки звучит зуммер окончания работы стиральной машины. Стиральная машина отличается от других бытовых аппаратов, поскольку за все время, исключая время работы, требуется, чтобы электрический ток был отключен. Поэтому после работы при помощи ОМЭВМ отключается самоблокирующее реле и прекращается подача электрического тока. Если во время работы открывается крышка машины, при помощи экстренной блокировки работа аппарата прерывается. Комбинируя предлагаемые программы стирки, можно выбрать одну из 63 вариантов. Установленная предварительно программа (6 минут стирки, 2 раза полоскание и 2,5 минуты отжимание) представляет собой «стандартную программу».

Структура контроллера для стиральных машин показана на рис. 13.

Входные переключатели — это кнопки сканируемой микро-ЭВМ клавиатуры, с помощью которой выбирается один из вариантов «программы по желанию» или «стандартная программа», а также переключатели уровня воды и переключатель, определяющий состояние крышки стиральной машины.

При работе микро-ЭВМ анализирует положение этих переключателей и показания таймера и в соответствии с программой, записанной в ПЗУ объемом памяти 768 байт, вырабатывает сигналы управления кремниевыми двунаправленными тиристорами Т, управляющими клапанами подачи и откачки воды, и изменением направления вращения ротора двигателя стиральной машины.

Микро-ЭВМ организует также управление звуковым сигналом окончания работы и самоблокирующимся реле, которое срабатывает после окончания процесса стирки или если переключатель крышки стиральной машины устанавливается в положение «открыто».

Режим работы машины отобра-

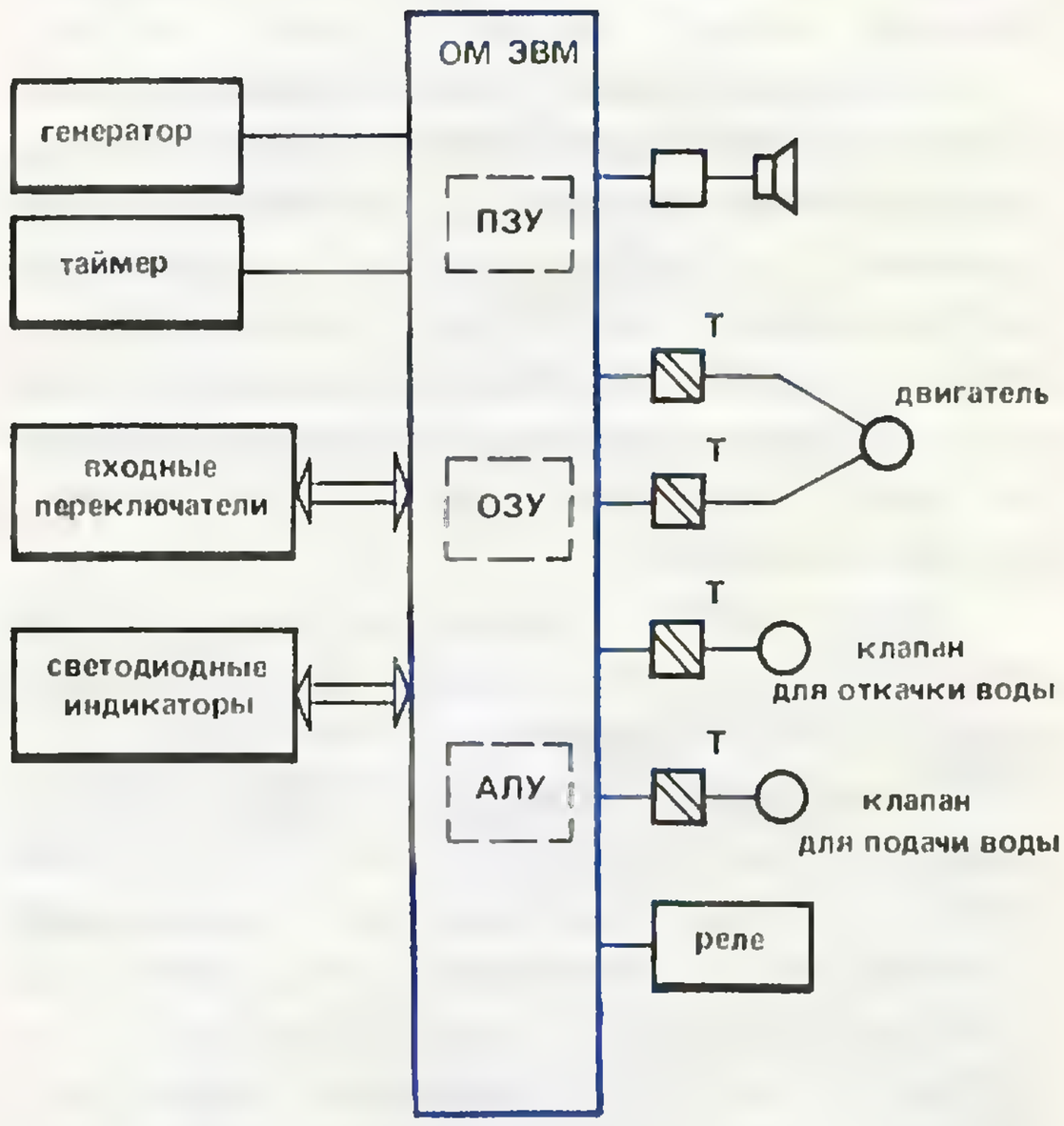


Рис. 13

жается с помощью индикаторов, выполненных на светодиодах.

При помощи покрытия специальными лаками узлов контроллера добиваются достаточной их защищенности от воздействия влаги и воды.

Контроллеры для холодильников

В прошлом для холодильников и рефрижераторов не выполнялась индикация температуры внутри холодильной камеры. В настоящее время широкое распространение получили холодильники для бытовых потребностей и большие промышленные рефрижераторные установки, поэтому с точки зрения экономии электроэнергии стало желательно осуществлять соответствующий контроль за температурой.

Управление такими холодильниками осуществляется с помощью контроллеров с микро-ЭВМ [Л2].

На рис. 14 а показана структура контроллера для холодильника. На рис. 14 б приведена диаграмма последовательности управления в контроллере:

непосредственно после окончания обмерзания возникает угроза повышения температуры внутри камеры, поэтому вентилятор охлаждения на некоторое время останавливается;

в управлении размораживанием контроль осуществляется по температуре, полученной в конце замораживания, и регулируется мощностью нагревателя для размораживания. Время таймера для размораживания корректируется в зависимости от того, высокая или низкая температура снаружи.

Микро-ЭВМ представляет собой однокристальную микро-ЭВМ с объемом ПЗУ 2 Кбайта.

Регистрация температуры внутри камеры выполняется на основании показаний температурных сенсоров, размещенных внутри морозильника и холодильника. В конструкцию контроллера входит 5-битовая сканирующая схема поиска и выбора температурных сенсоров. Аналоговый сигнал с выхода ЦАП поступает на входы компараторов и сканирует пять указанных сенсоров температуры. Управление процессом сканирования осуществляется аппаратно от ОМЭВМ.

В контроллере на основании показаний датчиков рассчитывается температура внутри камеры и вырабатываются необходимые сигналы для реле управления нагревателем для размораживания, управления компрессором и мотором вентилятора. Происходит также отображение полученной температуры морозильной и холодильной камеры с помощью люминесцентных индикаторов.

Микро-ЭВМ имеет также два ключевых входа, которые используются для синхронизации работы знаковых индикаторов в соответствии со сканирующими сигналами от микро-ЭВМ.

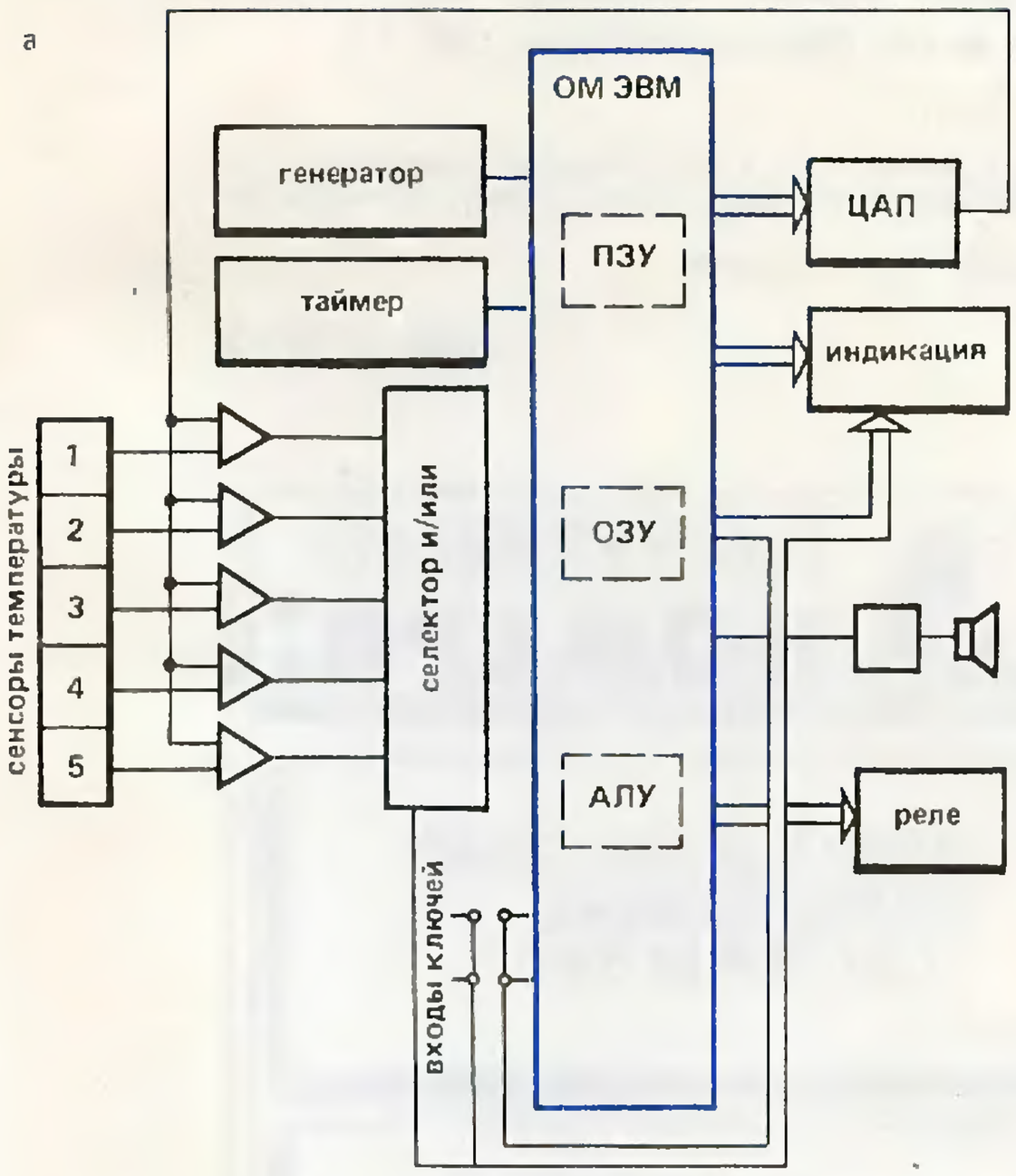
Если температура в камере оказывается за пределами заданной области, вырабатывается звуковой сигнал предупреждения, который заставляет оператора холодильной установки обратить внимание на температуру внутри камеры. Если в контроллере используются специальные дверные переключатели, предупреждающий звуковой сигнал генерируется и тогда, когда пользователь забывает закрыть дверцы холодильника.

В будущем предвидится тенденция к развитию разделенных систем, у которых компрессор будет размещаться в наружном помещении, а холодильные камеры под полом. Применение ОМЭВМ в таких системах позволит добиться большой эффективности с точки зрения экономии энергии и в смысле уменьшения объема компрессора.

Контроллеры для автомобиля

Вычислительные возможности ОМЭВМ позволяют использовать их для выполнения многих задач в автомобиле [1].

Одной из самых важных является задача управления двигателем автомобиля. Сюда включаются такие функции, как управление система-



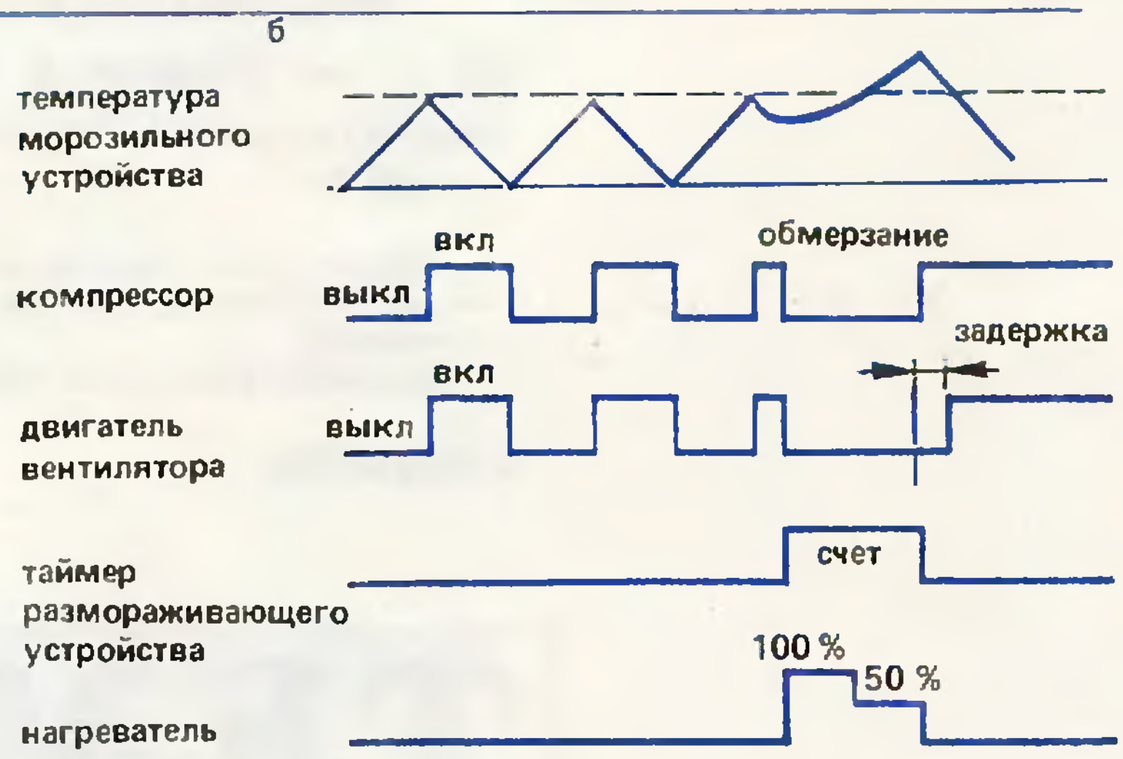


Рис. 14

ми, расходующими топливо, контроль токсичности выхлопных газов, цифровая настройка автомобильной радиоаппаратуры, выдача на дисплей показаний различных диагностических датчиков автомобиля и т. д. В результате обеспечивается экономия топлива, уменьшение механического изнашивания автомобиля, повышается безопасность управления автомобилем.

Применялся в автомобиле одноплатный микрокомпьютер COPUS-RUISE, построенный на основе ОМЭВМ 400, обеспечивающий выполнение 44 функций, основные группы которых: поддержание предварительно заданной скорости, информация о наиболее экономичном режиме, диагностическая информация. Можно задать перед поездкой расстояние, которое предстоит пройти, и на дисплей в течение поездки будет выводиться информация об оставшемся расстоянии, количестве топлива и времени поездки на основе данных о скорости и расходе топлива.

Примером применения отечественных микро-ЭВМ в автомобильных контроллерах является автомобильная бортовая микро-ЭВМ на основе ОМЭВМ серии К1820. Эта бортовая ЭВМ — стандартное устройство для выполнения ограниченного числа функций. Основное назначение ее — контроль расхода топлива.

При нажатии одной из функциональных клавиш микро-ЭВМ выдает следующую информацию:

мгновенное потребление топлива в литрах на 100 км или при скорости менее 20 км/ч — в литрах на час;

среднее потребление топлива в литрах на 100 км;

абсолютное потребление топлива от старта в литрах;

среднюю скорость в км/час (результат виден также после выключения зажигания и на остановках);

продолжительность поездки в часах и минутах, включая время остановок;

показания стартовых часов;

подсчитанное расстояние для режимов заднего хода.

Литература

1. Применение однокристальных четырехразрядных микро-ЭВМ К1820ВЕ2 и К1820ВЕ1. Информационно-справочное издание.

2. Маэда Т., Есиока Т., Окамото К. Контроллеры бытового назначения на базе 4-разрядной однокристальной микро-ЭВМ.

3. Бобков В. А., Бурмистров Ю. Н., Кособрюхов В. А., Уткин Ю. В., Чернуха Ю. В. Однокристальные 4-разрядные микро-ЭВМ серии КР1820.— Микропроцессорные средства и системы. — 1986. — № 1.

М59 **Микропроцессоры.** — М.: Знание, 1989. — 48 с. — (Новое в жизни, науке, технике. Сер. «Вычислительная техника и ее применение»; № 1).
20 к.

Рассмотрены вопросы истории разработки и производства микропроцессоров и микропроцессорных устройств. Приведены оригинальные примеры их применения.

Брошюра рассчитана на широкий круг читателей.

2302030700 **ББК 32.973.2**

ТЕМА *СЛЕДУЮЩЕГО* **НОМЕРА:**

РАДИО-ЭЛЕКТРОНИКА И СВЯЗЬ

НЕТРАДИЦИОННАЯ ЭЛЕКТРОНИКА (СИСТЕМЫ ЗУ)

***ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА* И ЕЁ ПРИМЕНЕНИЕ**

ТЕХНОЛОГИЧЕСКАЯ БАЗА ИНФОРМАТИКИ

МАТЕМАТИКА КИБЕРНЕТИКА

ВЫЧИСЛИТЕЛЬНЫЕ МЕТОДЫ ЛИНЕЙНОЙ АЛГЕБРЫ

Научно-популярное издание

МИКРОПРОЦЕССОРЫ

Гл. отраслевой редактор *Л. А. Ерлыкин*
Редактор *Б. М. Васильев*
Мл. редактор *Н. А. Васильева*
Художники *В. Н. Конюхов и К. Н. Мошкин*
Худож. редактор *М. А. Гусева*
Техн. редактор *Т. В. Луговская*
Корректор *О. А. Лагуненко*

ИБ № 9808

Сдано в набор 27.09.88. Подписано к печати 30.11.88. Т-21565. Формат бумаги 70×100¹/₁₆. Бумага офсетная. Гарнитура журнально-рубленая. Печать офсетная. Усл. печ. л. 3,90. Усл. кр.-отт. 8,45. Уч.-изд. л. 4,27. Тираж 66 090 экз. Заказ 2528. Цена 20 коп. Издательство «Знание». 101835, ГСП, Москва, Центр, проезд Серова, д. 4. Индекс заказа 894701. Ордена Трудового Красного Знамени Калининский полиграфический комбинат Союзполиграфпрома при Государственном комитете СССР по делам издательств, полиграфии и книжной торговли. 170024, г. Калинин, пр. Ленина, 5.

Цена 20 коп.

Индекс 70195

Адрес подписчика:

Лм 27 43

Издательство «Знание»

Подписная научно-популярная серия

ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА И ЕЁ ПРИМЕНЕНИЕ

Сравнивать микропроцессор Intel *80386 с первым микропроцессором — это все равно, что поставить рядом космический челнок "Шаттл" и первый аэроплан "Китти Хоук" братьев Райт.*

Рид Макманус

Наш адрес:
СССР,
Москва,
Центр,
проезд
Серова, 4